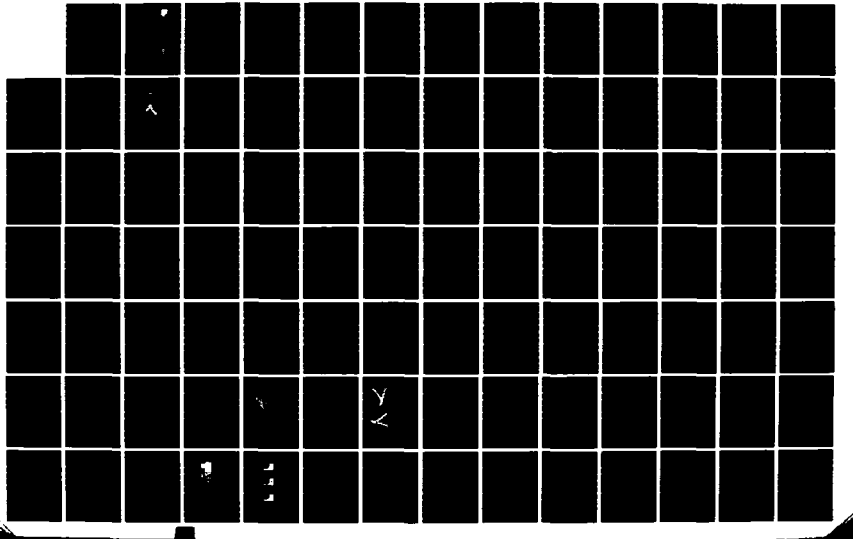
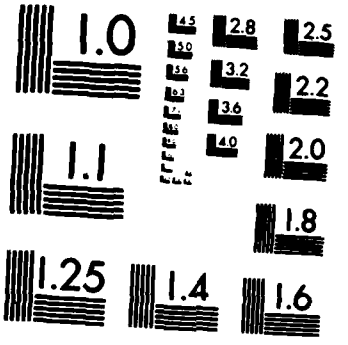
AD-A130 195 INTERACTIVE RASTER DATA STRUCTURE STUDY(U) SYNECTICS 1/3
CORP ROME NY D R GALLAVRESI ET AL. JAN 83
RADC-TR-83-12 F30602-81-C-0279

UNCLASSIFIED F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD A130195

RADC-TR-83-12
Final Technical Report
January 1983

(12)

INTERACTIVE RASTER DATA STRUCTURE STUDY

Synectics Corporation

David R. Gallauresi, James R. Muller, R. Patrick O'Connor and
David A. Kolassa

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DTIC
ELECTE
JUL 8 1983
S D
A

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441

DTIC FILE COPY

83 07 07 063

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-83-12 has been reviewed and is approved for publication.

APPROVED: John R. Baumann

JOHN R. BAUMANN
Project Engineer

APPROVED: John N. Entzminger

JOHN N. ENTZMINGER, JR.
Technical Director
Intelligence & Reconnaissance Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (IRRP) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-83-12	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) INTERACTIVE RASTER DATA STRUCTURE STUDY		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report 10 Nov 81 - 10 Nov 82
		6. PERFORMING ORG. REPORT NUMBER N/A
7. AUTHOR(s) David R. Gallauresi, R. Patrick O'Connor James R. Muller, David A. Kolassa		8. CONTRACT OR GRANT NUMBER(s) F30602-81-C-0279
9. PERFORMING ORGANIZATION NAME AND ADDRESS Synectics Corporation 111 E. Chestnut Street Rome NY 13440		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 45941737
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (IRRP) Griffiss AFB NY 13441		12. REPORT DATE January 1983
		13. NUMBER OF PAGES 287
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: John R. Baumann (IRRP)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Raster data
Automated cartography
Raster topology
Pixel connectivity

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

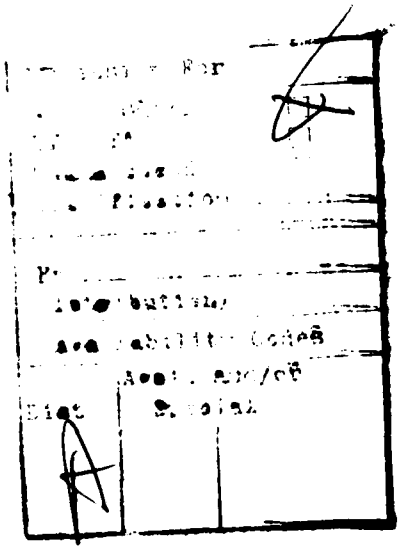
The report describes a study conducted to define current capabilities for interactive processing of raster structured cartographic data and to identify potential techniques for future enhancement of those capabilities. Past raster processing development activities are summarized and current concepts are described. Data structures are recommended to increase computational efficiency of various cartographic functions associated with the production and exploitation of topological data. A system

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

hardware and software concept is suggested to support future requirement for a complete raster compilation and revision capability.



DTIC COPY INSPECTED

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

APPENDICES

List of Illustrations

List of Illustrations (Continued)

List of Illustrations (Continued)

List of Illustrations (Continued)

List of Tables

SECTION 1. INTRODUCTION

This document is the final technical report representing a summary of the research performed for the Interactive Raster Data Structure (IRDS) study. This introductory section will present a background to the problems addressed by the study, acquaint the reader to the IRDS effort and will briefly discuss the organization of the report.

1.1 Background

In order to discuss the needs for an improved raster processing capability we must understand the forces of change and the trends toward the future which are prevalent throughout the mapping, charting and geodesy (MC&G) community.

The drive towards very large cartographic data bases to satisfy needs for product flexibility, timely responsiveness to user demands, and lower production costs is a phenomenon familiar to all suppliers of cartographic information. That is why advances in digital technologies have great potential for synergistic benefits to be gained by all members of the MC&G community.

A major obstacle to satisfying the need for digital cartographic data has been the high cost and long lead time associated with data capture of cartographic source material. Conventional digitization techniques using semi-automated cartographic systems simply cannot meet the heavy throughput required. Dramatic improvements in the rate of data capture and in the processing of cartographic data are promised by the emergence of raster technology within the digital mapping areas.

A number of influencing technologies have contributed to the continuing emergence of raster processing of cartographic data (See Figure 1-1). High resolution raster graphic displays such as multi-color CRTs, plotters and hardcopy recorders have continued to increase in performance and decrease in cost. Supplementing raster cartographic data gathered through input devices such as raster scanners and video "frame-grabbers" has been raster data acquired from other sources such as digital elevation models and digital remote sensor data. Computer hardware advances have offered increased memory capabilities for low cost mass storage and parallel processing architectures have increased raster data throughput. These trends will continue to offer increasing cost, quality and performance advantages for raster processing applications and lead to the expansion of raster-based cartographic systems not only with DMA but also throughout the MC&G community.

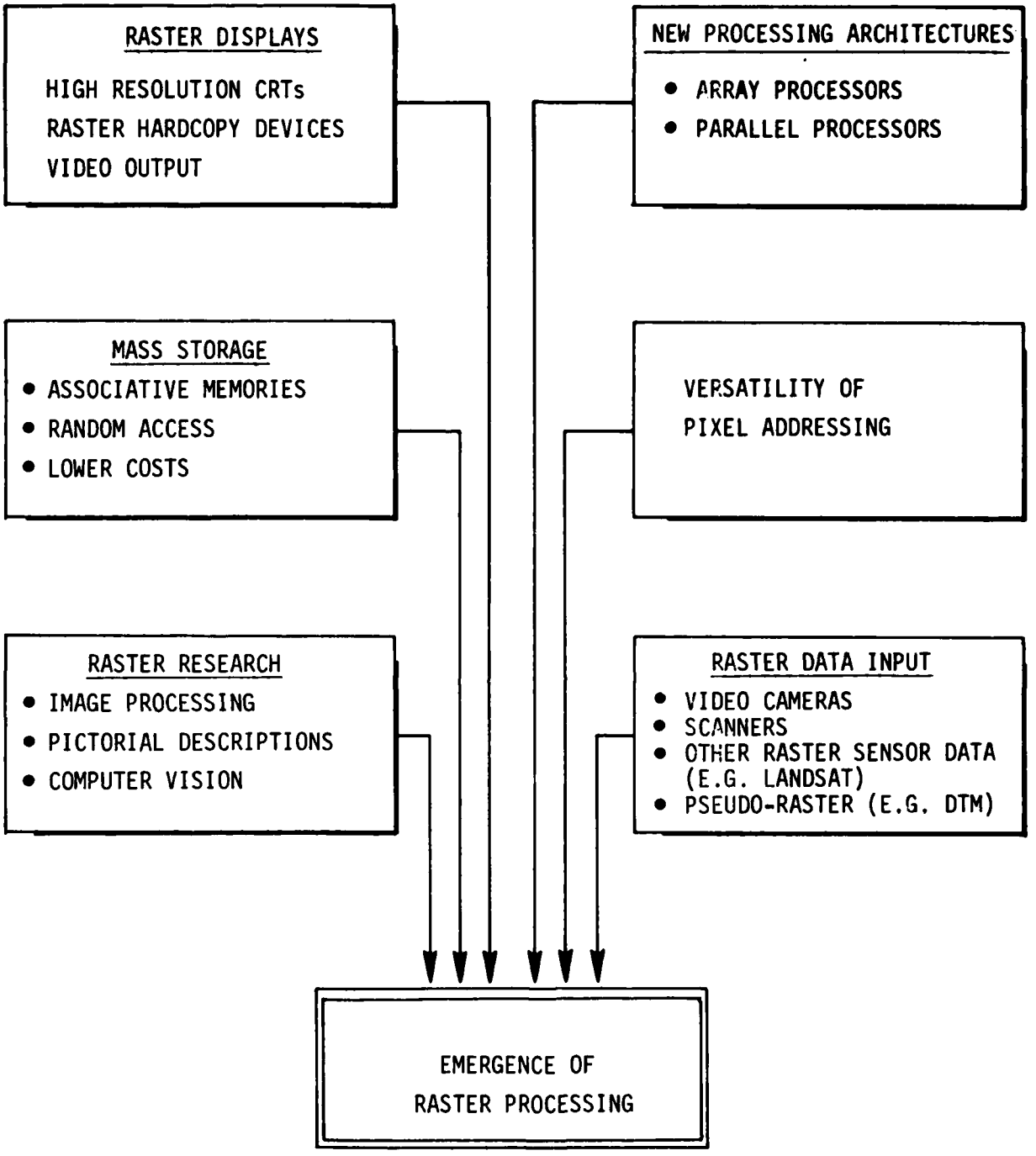


Figure 1-1, Technologies Influencing Raster Processing

A major deterrent to the acceptance of raster-orientated cartographic processing has been the strong tradition of vector-oriented digital cartography (See Figure 1-2). Early raster processing attempts required users familiar with lineal-based features to conceive of cartographic entities as scan lines and pixels of data. The relative maturity and widespread understanding of vector-based cartographic processing coupled with the lack of effective raster data handling and processing techniques initiated alternative efforts to convert raster data into conventional vector formats. Unfortunately the burden of time consuming raster-vector format conversions and the problems in the conversion processing offset the advantages of lineal processing of cartographic data.

1.2 An Overview of the Interactive Raster Data Structure Study

The IRDS effort was a one year, Air Force funded, 6.2 exploratory research study sponsored by the Rome Air Development Center (RADC). A project overview is presented as Figure 1-3. The requirements of the study were twofold:

1. Survey existing raster data structures; and

2. Design a data structure and algorithms to support interactive editing of raster cartographic data.

To accomplish these tasks the project team performed a comprehensive literature search examining the application of raster data structures in disciplines such as image processing, pictorial data description and computer vision. Secondly, discussions were held with the research components of: Defense Mapping Agency Aerospace Center (DMAAC), Defense Mapping Agency Hydrographic and Topographic Center (DMAHTC), Eastern Regional Remote Sensing Applications Center (ERRSAC), and Decision Information Display System, both of NASA/Goddard Space Flight Center, United States Geologic Survey, and U.S. Army Engineer Topographic Laboratory (ETL), and, of course, Rome Air Development Center.

The results of this preliminary research were reflected in an interim technical report produced midway through the project which served as a basis for the subsequent data structure design. The data structure design and the survey share the common purpose reflected in Figure 1-4., which is to provide an interface between the digital raster cartographic data and the feature orientated processing which is typical of cartographic editing. The interface capability allows cartographers to interact and manipulate the raster data in more familiar feature orientated terms while retaining a raster data structure.

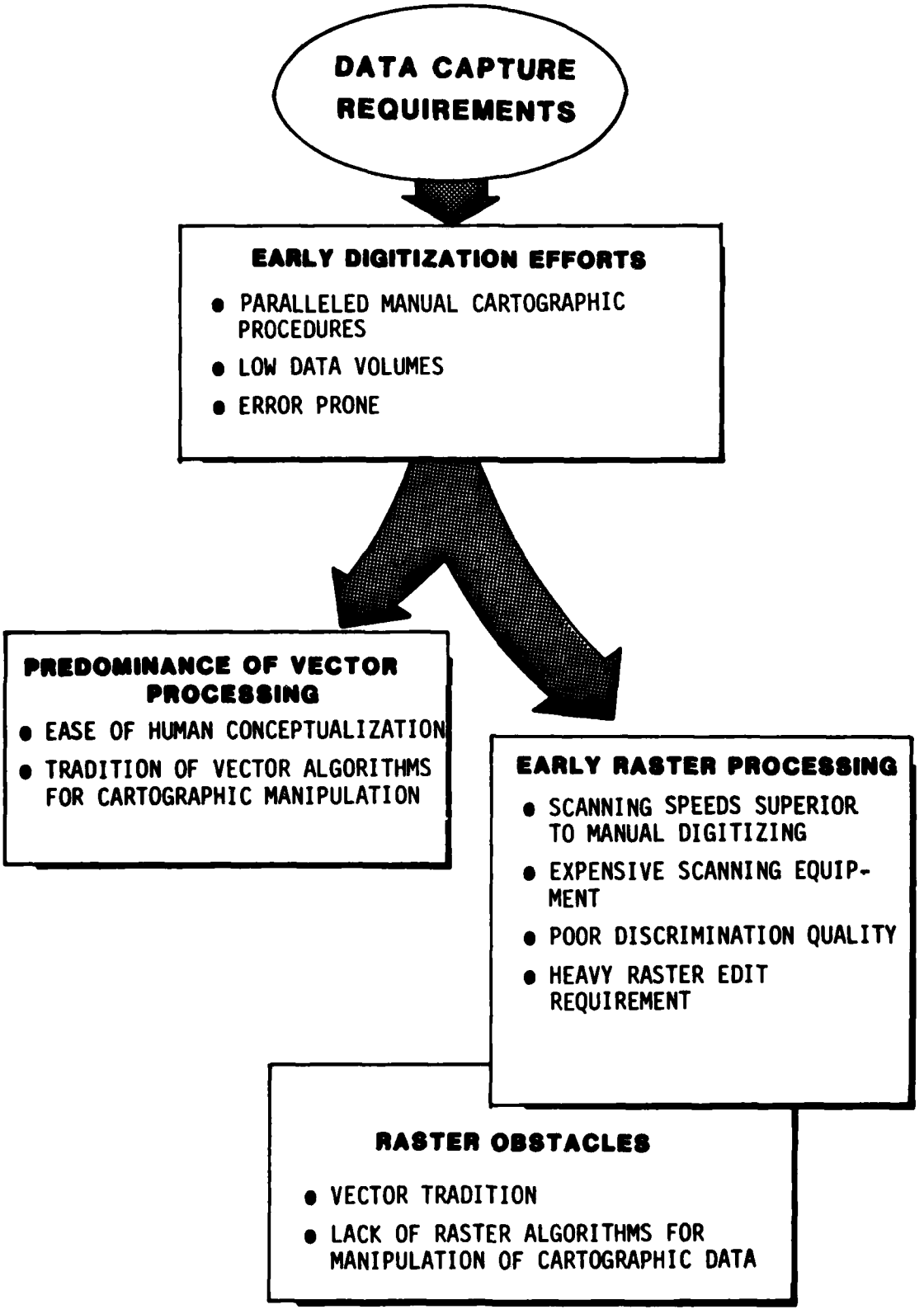


Figure 1-2. Cartographic Data Collection Methods

- 6.2 EXPLORATORY RESEARCH
- AIR FORCE FUNDED - ROME AIR DEVELOPMENT CENTER (RADC)
- NOVEMBER 6, 1981 - NOVEMBER 8, 1982
- PRIMARY TASKS
  1. SURVEY AND ANALYZE EXISTING RASTER DATA STRUCTURES
  2. DESIGN A DATA STRUCTURE AND ALGORITHMS TO SUPPORT INTERACTIVE EDITING OF RASTER CARTOGRAPHIC DATA

Figure 1-3. Interactive Raster Data Structures Project Overview

- IDENTIFY RASTER DATA PROCESSING TECHNIQUES SUITABLE FOR INTERACTIVE CARTOGRAPHIC PROCESSES

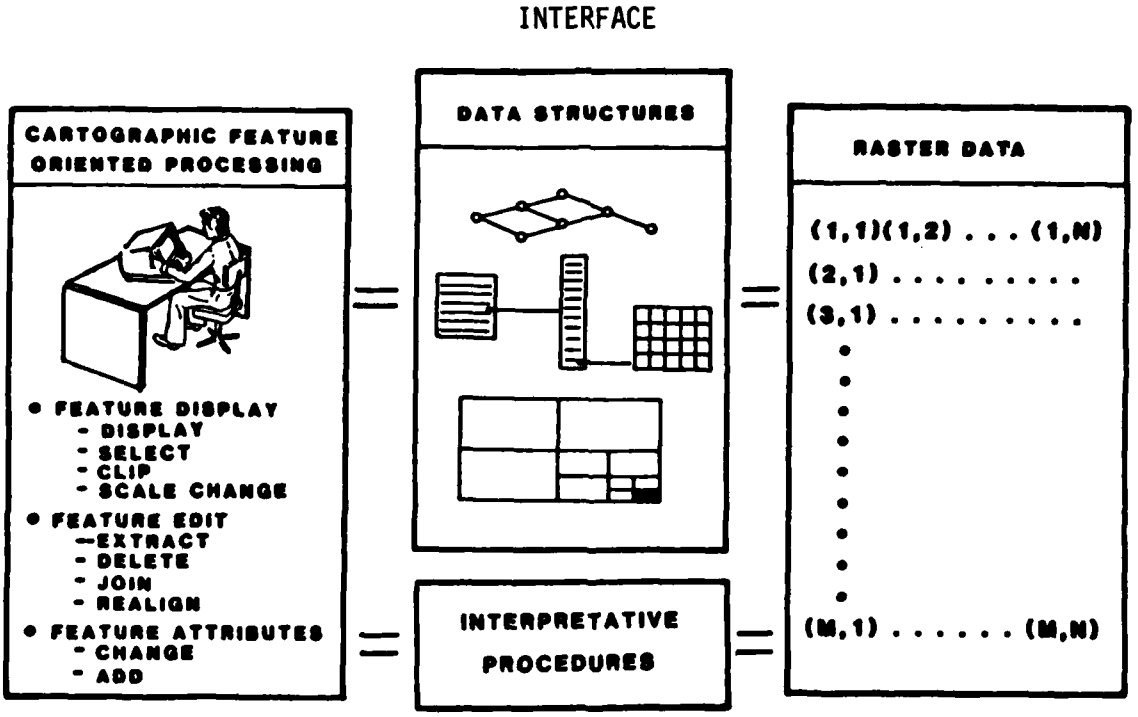


Figure 1-4. Purpose

1.3 Organization of this Report

This report is organized into nine sections, a bibliography, and three appendices. The report proceeds from the general to the specific. In addition to this introduction the material contained in each of the sections is highlighted as follows:

Section 2 - Executive Summary, outlines the particularly salient aspects of the report.

Section 3 - Background of Raster Processing in the Cartographic Environment, presents the emergence of raster technologies and their impacts upon the current cartographic environment. Past raster research efforts at both RADC and ETL are summarized and finally there is a presentation of the current approaches to processing raster data.

Section 4 - Survey of Current Raster Data Structure, describes many of the raster structures which were discovered through our survey effort. The discussion proceeds in a top-down fashion by describing information structures, logical structures and storage structures.

Section 5 - Study Goals, defines the desired IRDS objectives for a processing environment in terms of cartographic functions, exploitation of topological data properties, exploiting parallelism and increasing both human and computational processing efficiency.

Section 6 - Data Structure Elements, describes the properties of the recommended three elements of a raster data structure; pixel value, pixel connectivity map and connected component label.

Section 7 - Concept of Operations, presents a software orientated overview of a raster processing flow through a cartographic compilation system, including the creation of the data structure elements, and the cartographic manipulations which use them.

Section 8 - Support Environment, takes an essentially hardware view of the tools which are necessary and desirable to support the processing flow. The section revolves around a discussion of the components of a full scale raster compilation and revision system and a discussion of applicable parallel architectures.

Section 9 - Summary and Recommendations, synopsizes the report and describes what should be done to expand this work towards a production capability.

APPENDIX A - List of Algorithms, A listing of a recommended connected component labeling algorithm.

APPENDIX B - Data Structure Evaluation Technique, discusses the factors which influence both measurable and perceived performance of data structures and walks through an example evaluation of alternative data structures and algorithms for contour filling.

APPENDIX C - Relevant Data Storage Technologies, contains more detail about the particular hardware technologies recommended in Section 8.

SECTION 2. EXECUTIVE SUMMARY

This section contains a brief synopsis of the entire IRDS technical report. It is assumed that the reader has previously covered the material presented in Section 1 Introduction, in order to gain the necessary background to the IRDS effort. Of particular importance are Section 1.2 An Overview of the Interactive Raster Data Structure Study, and Section 1.3 Organization of this Report. These two sections briefly cover the purpose and scope of the study and describe the contents of this report.

The key topic addressed by this report is the presentation of a raster data structure suitable for cartographic editing of raster data. This data structure is founded upon a wide ranging literature search which investigated raster data structures in varied applications. This research encompassed over one hundred references and reported upon over 25 diverse data structures used for storage and manipulation of raster data; much of this work is summarized in Section 4 Survey of Current Raster Data Structures. The study of these data structures, their purpose and the properties which they incorporate helped us establish objectives which should be addressed by raster data structure to be used in a cartographic editing environment. Simply put these objectives are:

1. The structure should facilitate parallel processing techniques.

2. The data structure should support the full range of functions which exist in the cartographic editing environment.

3. The structure should incorporate the topologic properties of adjacency and connectivity to the greatest extent possible.

4. The structure should enable interpretations to be applied via software techniques to determine data features which were formerly part of a user's decision making. This enables off loading of decision making and detailed specifications into the software.

5. Foremost the structure should be simple to create and update. In a cartographic editing environment the data to be used is very temporal and dynamic in nature, Changes in the data being addressed should not impose a burden upon the processing flow.

These objectives are further discussed in Section 5 Study Goals and are met by the data structure proposed in Section 6 Data Structure Elements. The raster data structure incorporates three elements which meet the aforementioned goals:

1. Pixel Value - a numeric code which represents a color, grey scale level or pixel intensity.

2. Pixel Connectivity Map (PCM) - is an eight bit code in which select bits are set to zero (0) or one (1) to indicate the presence of adjacent and connected similar surrounding neighbors.

3. Connected Component Label - is an identifier which uniquely distinguishes a group of connected pixels which may form a subset of a feature.

The pixel value or color code is the current basis for manipulation algorithms in raster systems. Processing and manipulations are enhanced by the Pixel Connectivity Map (PCM). The purpose of the PCM is to reduce the computational burden of accessing and processing local neighborhoods. Use of an eight bit PCM subdivides the computational burden into a single accessing of a pixel's eight neighbors, assembling them into the PCM and then computing the desired neighborhood function by simple lookup and masking operations on the PCM. A set of 26 predefined bit mask templates can be applied using simple logical comparison operations (AND, OR, NOT) to uniquely classify raster elements as:

1. Interior Pixels;

2. Contour Pixels;

3. Arc Endpoints;

4. Local minima and maxima;

5. Minima and maxima of holes within a connected component; and

6. Junctions or intersections between connected components.

Finally, the connected component label serves to aggregate individual pixels into subsets of connected pixels. Use of the PCM allows multiple definitions of pixel connectivity to be applied; directly connected (i.e., side touching), indirectly connected (i.e., corner touching) and spatial segmentation at X and T junctions by splitting connected components at detected intersections.

The creation of these particular data structure elements and the design of the manipulation techniques to utilize them in an editing environment is discussed in Section 7 Concept of Operations. The software operations include:

1. Segmentation;

2. PCM Encoding;

3. Connected Component Labeling;

4. Cartographic Manipulations; and

5. Merge/Conflict Resolution.

These functions form an iterative cycle in which the first three steps are done to support the manipulations and the last step assists in "cleaning up afterwards".

One of the powerful aspects of the raster data structure is its ability to take advantage of current raster display technologies, specifically the increasing use of local high speed RAM memories. This capability facilitates implementation of raster editing stations in a distributed station environment where many highly interactive stations can be serviced by a host machine managing the data bases. A full discussion of a possible hardware support environment is presented in Section 8 Support Environment, accompanied by a presentation of the use of a parallel processing architecture to perform many of the software functions.

In summary, this report presents the supporting research for the IRDS data structure design, the raster data structure design itself, a detailed discussion of raster editing manipulation techniques based upon the data structure and conjecture regarding hardware support environment. What remains is the translation of these results into a production environment. This can only be accomplished by implementing and testing the discussed techniques in software and hardware. To move from the presentation of ideas and development of techniques to a production capability requires work in a testbed environment to explore alternative strategies to ensure an easy to use, and efficient implementation.

SECTION 3. OVERVIEW OF RASTER PROCESSING IN THE CARTOGRAPHIC ENVIRONMENT

The need for automation in the cartographic production environment has been widely accepted. In the past, chart production relied on manual methods which required time and specialized skills. With the advances in computer technology the means to support and automate the procedures became available. Consequently, many universities, research centers, and commercial firms became involved in exploring new technologies which could support automated cartography. Raster technology emerged from early investigations as a plausible alternative and continues to be a valid component of automated cartography programs.

Valuable knowledge has been gained from past and current raster efforts from which to base future research directions. The following sections provide a brief overview of how cartographic raster processing emerged along with some specific accomplishments that have been made within the Department of Defense. The section is concluded with a look at the current approaches and problems of raster technology.

3.1 Emergence of Raster Processing in Cartography

Early digital cartographic efforts began by paralleling the manual cartographic process. Features were digitally recorded as vectors or lineal chains in the same way a cartographer would draft them. This allowed for the natural conceptualization of digital data as features.

This trait has been observed as a result of functions being implemented by those trained in manual methods; and the hardware at the time, was readily adaptable to the replication of manual methods (Peuquet, 1976).

Although automation had been introduced, data capture was slow and error prone due to the necessity of human intervention. Consequently, efforts were launched to develop new methods and technologies which might offer a more productive alternative. The result was the identification of raster technology to rapidly extract digital data from analog sources. Thus, early cartographic raster processing was founded on the development of raster scanning hardware (See Figure 3-1).

The early sixties produced several raster scanners which were able to sample cartographic source material and digitally record the presence or absence of cartographic features. Initial applications of the output focused on map duplication purposes. Foremost was the raster digitizing of cartographic manuscripts to produce color film separations (Palermo, 1971; Clark, 1980). Data capture rates were significantly

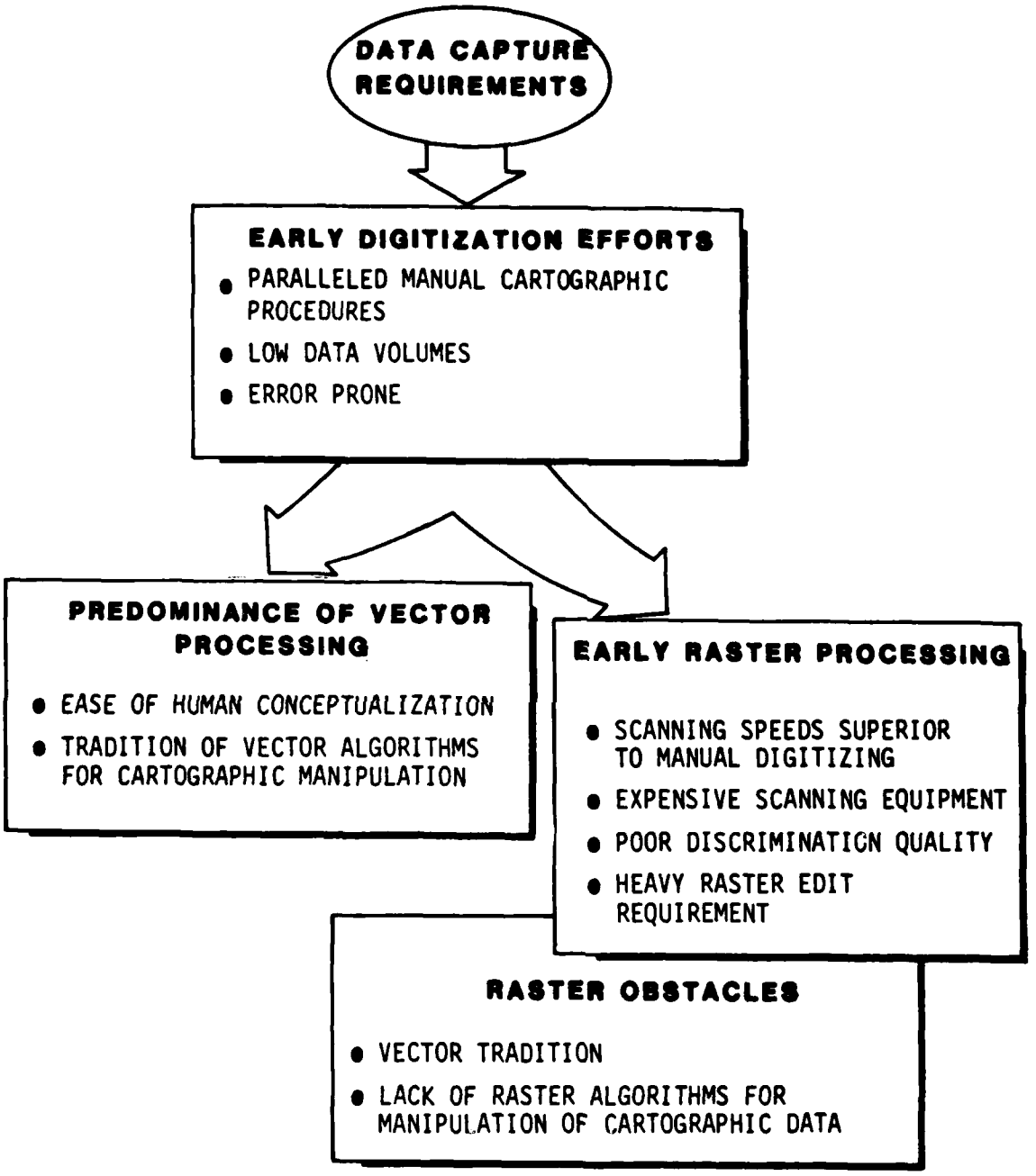


Figure 3-1. Early Data Capture Process

higher than manual digitizing but were offset by the expense of early scanning equipment and cumbersome editing procedures.

A major obstacle in the use of raster data for other applications was the inability to perform feature or segment manipulations on raster elements in a fashion familiar to most cartographers. In an attempt to solve this problem several efforts were made to convert raster scanned data into the familiar representation of vector chains. The raster to vector conversion processes typically encountered two problems; a high processing overhead; and the difficulty of associating attributes with newly created vector chains.

3.2 RADC Efforts

RADC's efforts to automate cartographic processes began with their development of the Advanced Cartographic System (ACS). As part of the ACS, RADC initiated the development of the Automatic Color Separation Device (ACSD). The basic task of this device was to convert color-coded graphics into digital form and to record it on magnetic tape. Color was detected and converted to a format which defined the color encountered, the location, and whether it was a line or area information. This device performed these operations based upon raster scan principles and employed a rotating drum to provide positional data in rectangular coordinates. Some problems were encountered in subsequent processing of lines that ran parallel to the motion of the drum. However, the system was believed to have provided a significant advancement to the cartographic community.

Following the completion and installation of the ACSD, a study was undertaken to thoroughly analyze and exploit this new device. The Computer Assisted Scanning Techniques (CAST) study was performed to determine a process that could achieve the goal of using raster scan raw data in an operational environment. The study was specifically interested in the absolute performance of the ACSD and the effects of mechanical settings. In addition, the effort required an experimental software system whose raster to lineal conversion would provide a suitable graphic output.

To explore the raster to lineal conversion, software was developed and implemented on a PDP 9 under the CAST effort. The principle of the linealization technique employed was to establish relationships between disconnected data points within an array formed by the raster points of three consecutive scan lines. Correlations between data points were made within a neighborhood about a given point on the scan line using predetermined criteria for establishing distance and direction between that point and the next. Line segment lists were constructed, linked, formatted, and then ouput. This system did not perform very well due to deficiencies in the algorithm which resulted in feature segmentation.

and hardware limitations and allowed limited amounts of data to be processed at one time. However, it was felt that the ACSD parameters were fairly well understood and software problems in the conversion process had been isolated.

The completion of the CAST study brought out continued interest to further develop the ACSD. The Raster Plotter Software project was begun with two prime objectives. First, a hierarchical color coding scheme was needed to represent cartographic feature classifications. This was addressed by the Feature Identification Software System (FISS) as part of the Raster Plotter Software effort. FISS consisted of stand-alone programs which accepted ACSD generated raster data and then applied the hierarchical color scheme. Secondly, the project was tasked to implement software for applying line weight symbology to edited raster data for output on a plotter. The problems encountered were in the areas of the color coding scheme. In order to work properly, it required features to be in a lineal format in a continuous manner. The raster to lineal conversion techniques available at the time provided segmented representations of features.

During the same time period, an investigation into information displays resulted in the Color Scan Converter System (CSC). It represented a technological advance in low cost, high performance, color digital to video display devices. The CSC provided a one thousand line, multicolor raster type display of computer generated alphanumerics and line data. This effort is an interesting note because it demonstrated RADC's interest in other aspects of the raster technology.

The next major effort was the Computer Assisted Color Scanning project (CACS). This system aimed at refining the feature extraction and conversion used techniques similar to the CAST effort but with some significant enhancements. Implemented on a GE 635 mainframe, CACS was able to contain in core more consecutive scan lines simultaneously. This provided a better capability to deal with wide area data. Several special purpose routines within the system were available to recognize and evaluate special problem areas. Although the lineal output from CACS, in most instances, showed improvement over previous efforts, the error rate was still high.

The next stage in raster processing research involved further development of software that could accurately handle the lineal and raster conversion. This time frame also saw the implementation of systems employing raster scan technology in operational environments. These efforts are summarized as follows:

1. Color Detection Processing was aimed to enhance raster data products used as input to the lineal conversion process. Color discrimination techniques and data editing procedures were implemented and analyzed.

2. Raster-Lineal Editing Software was developed which would allow editing of cartographic data in raster format.

3. Raster Imaging Software was developed to support raster plotting capabilities at RADC. This included interactive point symbol and text manipulation and lineal to raster conversion of data for plotting purposes.

4. The Raster to Lineal Conversion Analysis resulted in a set of algorithms and techniques to further refine the conversion process. Employing the technique of line skeletonization, the software was implemented on a PDP 11/45, an H6000 and a UNIVAC 1108. This system could accept input tapes from various scanning devices and correctly linealize and raster scan data of good quality.

5. The Flip Raster Processing System installed raster imaging software on a minicomputer. This effort implemented a stand-alone system capable of supporting the production of Flight Information Publication graphics.

6. The Raster Plotter Software Improvements developed software necessary to generate tape drives for the raster finishing plotters in use at DMA. Conversion of lineal data to raster format was involved and produced output to create cartographic color separation negatives.

The period from the middle 1970s to the present witnessed a renewed emphasis into systems to support the cartographic production environment and provide interactive manipulation of data. The Color Raster Scanning System for Digitizing Cartographic Data was an intense effort to bring advanced technology to the cartographic environment. The effort resulted in a system composed of functional elements in hardware, software, and operator interfaces. The scanner that was built is capable of meeting many color raster scanning requirements with various input materials and different data extraction objectives.

The Raster Processing Techniques II developed techniques and algorithms to automatically track cartographic feature symbols in raster format and process the associated data. Three symbolized feature classifications were addressed: railroads, trails, and cased roads. Interactive intervention was necessary at feature intersections and along highly convoluted paths.

At the same time, the Lithographic Scanning Technique was initiated to develop techniques and algorithms to extract, identify, manipulate, and digitally record features from multicolored lithographic charts. The goal of the effort was to provide a capability to rapidly acquire color separation plates from existing foreign charts where original color plates are not available.

These efforts were followed by several projects still related to the cartographic environment. The Raster Scanned Character Recognition effort developed a system which used raster scan technology to create bathymetric data as input to a digital data base. Hand written numerical data was scanned on smooth sheets and processed to produce a digital output.

Standardizing raster data formats to provide convenient transfer between computer systems was the intent of the Analysis of Raster Data For Magnetic Tape Formats. A recommended standard was developed after an investigation of existing raster data formats was completed.

The most recent software development effort was the Raster to Lineal Conversion Software project. Raster To Lineal Conversion Analysis software was analyzed to identify and correct the problems. This effort also designed and implemented output modules to provide multiple output formats required to edit manuscripts.

3.3 ETL Efforts

An early raster hardware effort at ETL was the development of the Cartographic Scanner Plotter during 1971 to 1972. The objective of this effort was to design and build hardware capable of scanning and plotting cartographic information on the same unit (Clark, 1980). The device is able to scan in a raster format, analog sources that have been mounted on a rotating drum. As the drum rotates, a moving scanner head, incorporating a reflected light and optical system, detects pixel information. The information is then converted to a digital format, processed and output to high speed films. The resolution of the output is the same as the scanned data which could be 1, 2, or 4 mils.

The output can then be used to plot high quality color separation negatives. The tests performed under this effort found that extremely high quality maps could be printed from the color separation negatives produced. The primary disadvantage to the device was the need for total darkness to handle the high speed films.

Upon completion of the Cartographic Scanner Plotter, ETL began development that would support the hardware. Early software efforts were geared towards the USAETL CDC-6400 computer (Clark, 1980). Processes that were implemented included line thinning, vectorization, and conversion

of vector to raster. These early efforts uncovered many of the problems associated with the use of raster data. Testing that was conducted found that the processing of even simple images required an extensive amount of time to process. As a result, a different approach to the problem was sought which could increase the speed of processing raster data.

The approach taken by ETL in early 1973 was to use the STARAN associative array processor. This high speed processor is used as a peripheral processor on the ETL/CDC 6400 computer. This device was selected because it permits multi-dimensional access to all data stored in a series of arrays. The data is addressable by content rather than by coordinates. The STARAN associative array processor proved successful as the processing speed of raster data was greatly improved. Some of the efforts that were developed on the STARAN were (Barron, 1981):

1. STARAN Raster Processing System (STRAPS) developed raster to vector and vector to raster conversion programs. Line width detection and separation software as well as the generation of line and point symbols were accomplished under this effort.

2. Contour Digitizing and Tagging Software (CONTRAGRID) completed in 1979 demonstrated the capability of automated elevation tagging and gridding operations. Contour lines are taken through a raster-to-vector conversion, tagged for elevation, and then assembled into a uniform elevation grid.

3. Feature Tagging, as completed in 1980 had investigated algorithms which could recognize cartographic symbols. Most of the work centered on recognition of dual cased roads, railroads, depression contours and broken line symbology.

Other software efforts involve improvements to the STRAPS vector to raster conversion software. The symbolization process will also be studied for possible improvements.

One of the most current efforts at ETL has been the development of the Laser Platemaker (LPM) for digital mapping applications (USAETL, 1980). Digital data in raster format at high data rates are used by the LPM to write directly on to lithographic plates. This device will eliminate the need for producing and processing film negatives used in printing cartographic information on the lithographic plate. The savings in materials, time, and labor will be substantial when the LPM is fully implemented.

3.4 Current Approaches

The research to date has developed two distinct approaches for the utilization of raster data for cartographic purposes. The first, and most common, is seen in the raster to vector conversion systems. These systems usually employ a process of skeletonization to reduce lines to a single unit of width upon which line extraction and topology reconstruction is performed.

Conversion techniques insert an intermediate step prior to performing cartographic manipulations. In the compilation and revision environment where data is temporal in nature, additional processing leads to a corresponding decline in overall system productivity. While improved, the current raster to vector conversions still maintain some amount of processing overhead and encounter problems with:

1. Attribute tagging

2. Line coalescence in high density areas

3. Line gaps and variable width lines

4. Processing times highly dependent on data resolution

Furthermore, cartographic raster to vector conversion systems have primarily focused on contour and polygon type applications. This can be attributed to the specific rule sets which have been developed for these entities. More importantly, while the data is easily manipulated in vector format, its presentation to the user is foreign to its map symbology. During the conversion process, the symbology is removed leaving the user to interact with and manipulate data which is graphically different from the analog source or final product.

Contrasting the conversion systems are the all raster based systems. These are characterized by their retention of the raster format throughout the processing cycle. Systems utilizing this approach are able to offer the advantage of rapid data capture allowing the user to quickly initiate the manipulation functions. However, this approach has also been stalled by problems with the manipulation of data as raster elements. Operations are usually performed on a color or pixel basis which does not offer the same manueverability as the conversion systems.

In some situations these techniques may actually hinder the manipulation function. For example, the cartographic manipulation function of joining two features (See Section 4.1), can be very difficult to execute in a raster mode. Performing this function with a raster to vector conversion system, a user would need only to identify the features involved and issue

the join command. The endpoints of the features already exist as a terminal X,Y coordinate position, allowing the system to easily calculate the path the line should take and the point of intersection. Symbology would be added at a later step to graphically represent the features in their true form.

Raster data, however, does not deal with single pixel width lines. In actuality, it might be more accurate to describe lineal raster features as regions. This presents the problem of determining what pixel or group of pixels represent the endpoint of the feature. Once this has been determined, a judgment is required as to what direction it should take and how wide the connecting path should be.

This example demonstrates how the all raster approach requires several steps and extensive user guidance to perform a specific edit. The raster-to-vector conversion system would use simple mathematics and much less human interaction to perform the same task. Reasons such as this have led to the predominance of conversion systems.

SECTION 4. SURVEY OF CURRENT RASTER DATA STRUCTURES

The following section represents the bulk of the research performed under the IRDS study. Although the importance of the following data structures is far less than the concept for an all raster system, valuable information has been obtained. Investigating these data structures has provided a base from which to learn, conceptualize, and uncover important issues which might otherwise be overlooked. A point from which to compare and evaluate various structures has also been supported by the knowledge acquired during the effort. For these reasons the following data structures are included in this report.

4.1 Multiple Levels of Data Organization

Organization and representation of spatial data can be partitioned into discrete levels by examining the degree of data abstraction. Figure 4-1 is patterned after Nyerges (1980) and descends from a high level representation of data as ideas to the lowest level representation, machine encoding of the data. The remaining discussion will define these levels, provide an example and identify those of direct interest to IRDS.

At the highest level, labelled data reality, cartographic data exists as an idea which can be communicated among users of the data. For example, a cartographer and a planner can discuss geographic features represented on a map.

The canonical structure level is a loose organization of cartographic entities and associations which existed as ideas at the data reality level. This would correspond to the perspective at which a user group views the data.

At the information structure level, spatial data is conceived as cartographic entities, attributes of those entities and relationships among entities. This conceptual organization may be embodied in a data model or conceptual DBMS.

At the data structure level, data elements are represented in a logical fashion. Cartographic features, line segments, points, polygons, and grid cells are all logical expressions of particular data elements.

The storage structure level addresses the physical data formats and linkages to allow the above logical descriptions. Issues at this level are the degree to which logical structures are computed or encoded, data accessibility and the physical sequence of data fields and records.

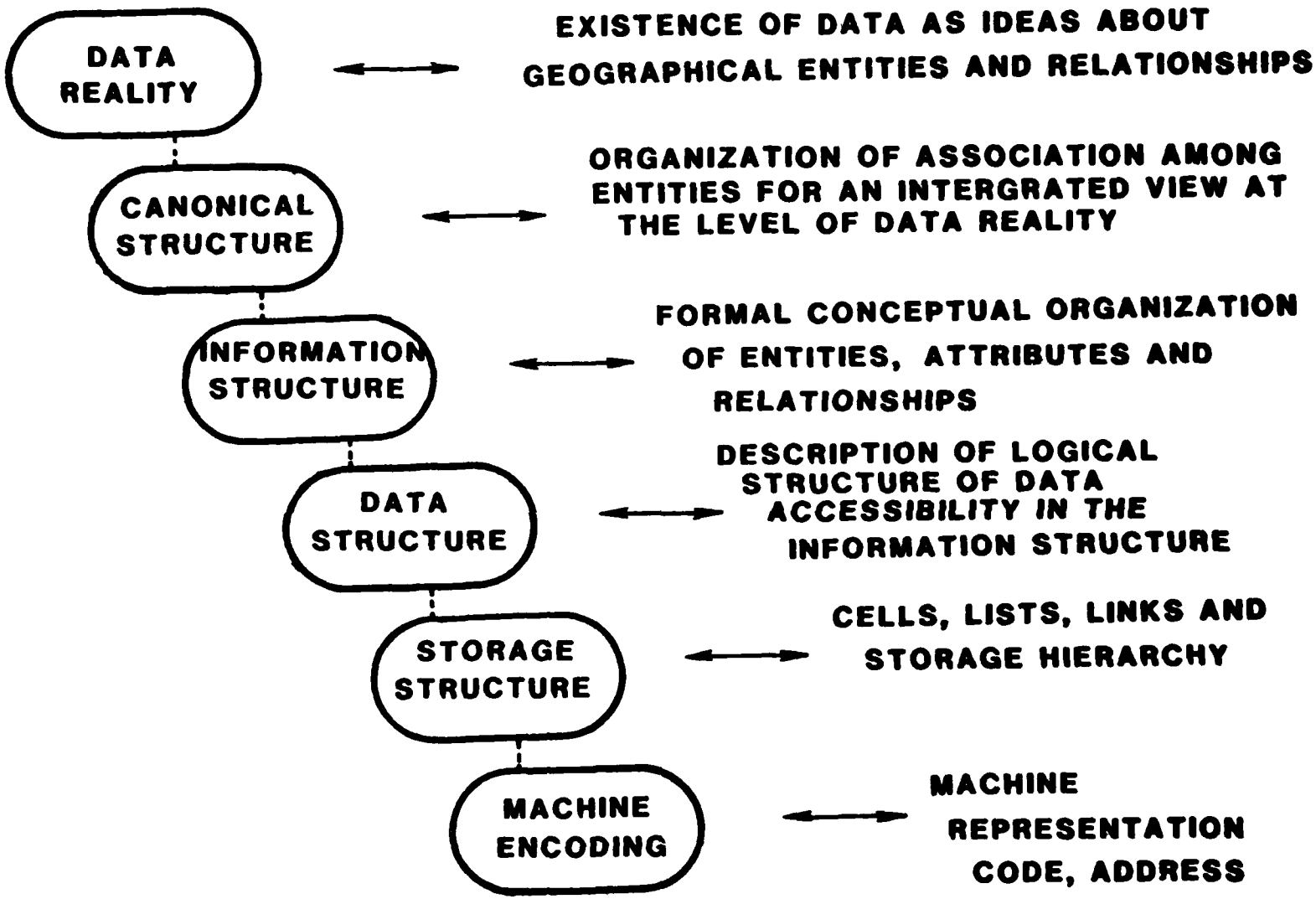


Figure 4-1. Multi-Level Models of Data Organization

Finally at the lowest level on the structure, the actual encoding of data occurs on a physical storage medium. Issues at this level are the addressing scheme used to store and retrieve data and the encoding schemes used to represent the cartographic entities, attributes, and relationships.

The scope of the IRDS study is primarily concerned with the information structure and storage structure levels which will support the data structure level. Information structure will be presented relative to its use in organizing of cartographic data. Storage structure is as to the techniques which will physically express all of the logical descriptions deemed necessary at the information and data structure levels.

4.2 Information Structure

An information structure as defined by Nyerges (1980) is a formal data model that specifies the conceptual organization of entities, attributes, and relationships. The general categories of entities and the types of relationships which exist between these categories are included within the information structure. We distinguish information structure from data structure in that the former is a scheme for representing knowledge. The latter is viewed as a description of the logical data access paths within the information structure. Information structure is a vital ingredient to support intelligence interaction, cartographic data understanding and cartographic production capabilities. Data structures or data access paths depend on explicit links such as in the case of tree or plex network models or reference tables as in relational models.

The Association for Computing Machinery (ACM) and the American National Standards Institute (ANSI) and other committees (i.e., CODASYL, SHARE, GUIDE, etc.) have fully addressed the subject of data base organization. The consensus is that the overriding objective of determining the logical structure of data is to ensure that the data will serve future diverse applications. Since its future uses cannot be totally foreseen the structure should be designed in order to represent the true properties of the data. The associations between data items should represent the inherent structures of the data (Martin, 1977). Thus, while the uses of the data may change and their physical structures may change, the inherent logical structure will remain stable and return its validity. In the next subsection we discuss the information structure of map data based on the organizational principle of representing the data in terms of its true properties.

4.2.1 DATA MODELS

Data models are mappings that show the logical association between data entities and their logical representations (Martin, 1977). The

three most common models are hierarchical, network, and relational. The hierarchical structure represents one-to-many relationships; network structures can represent many-to-many relationships; and relational structures employ a multiple-table representation without explicit connections between the tables (reference Figure 4-2). There is ample evidence that these three models are generally deficient for cartographic data handling, particularly with respect to representing spatial information and performing spatial operations.

Nyerges (1980) points out some of the problems with the network model. The most serious of these are the inabilities to distinguish between:

1. Spatial relationships and entities

2. Logical relationships which are part of the cartographic syntax and spatial or phenomenological relationships which are semantical in nature.

Also there is some doubt as to the ability to achieve data independence at the information structure level.

Hagan (1981) has done considerable research and development in the application of network models to cartographic data structures. In her work, she was restricted to using hierarchical CODASYL DBMS which permits only one-to-many relationships. Hagan successfully introduced linkage records to permit node to segment relations, segment to polygon relationships and other many-to-many relationships to achieve the network model. She was able to achieve sufficient flexibility in the structure to preserve spatial properties and support cartographic description by representing these two types of information in two distinct levels. The high level contained the semantical descriptions of features with the human interpreted meaning. The lower level contained the syntactical representation of nodes, segments and polygon coverage. Therefore, many redundant polygons which have no meaning in the map representation were introduced in order to avoid searching segment records. The evaluation of this network data structure with respect to complex data manipulation and operations was not complete.

Relational models represent relations between unordered sets of attribute values and are expressed in a tabular form. Each relation (table) consists of tuples (rows) and attributes (columns). Each tuple describes an entity (object). Relations achieve a high degree of data independence because the tabular form is an access-path independent representation. Relations have been used in a hierarchical manner to represent structural (syntactic) relationships in pictures and images. Relational models have also been used to represent semantic relationships by use of abstractions known as aggregation (higher level entity defined

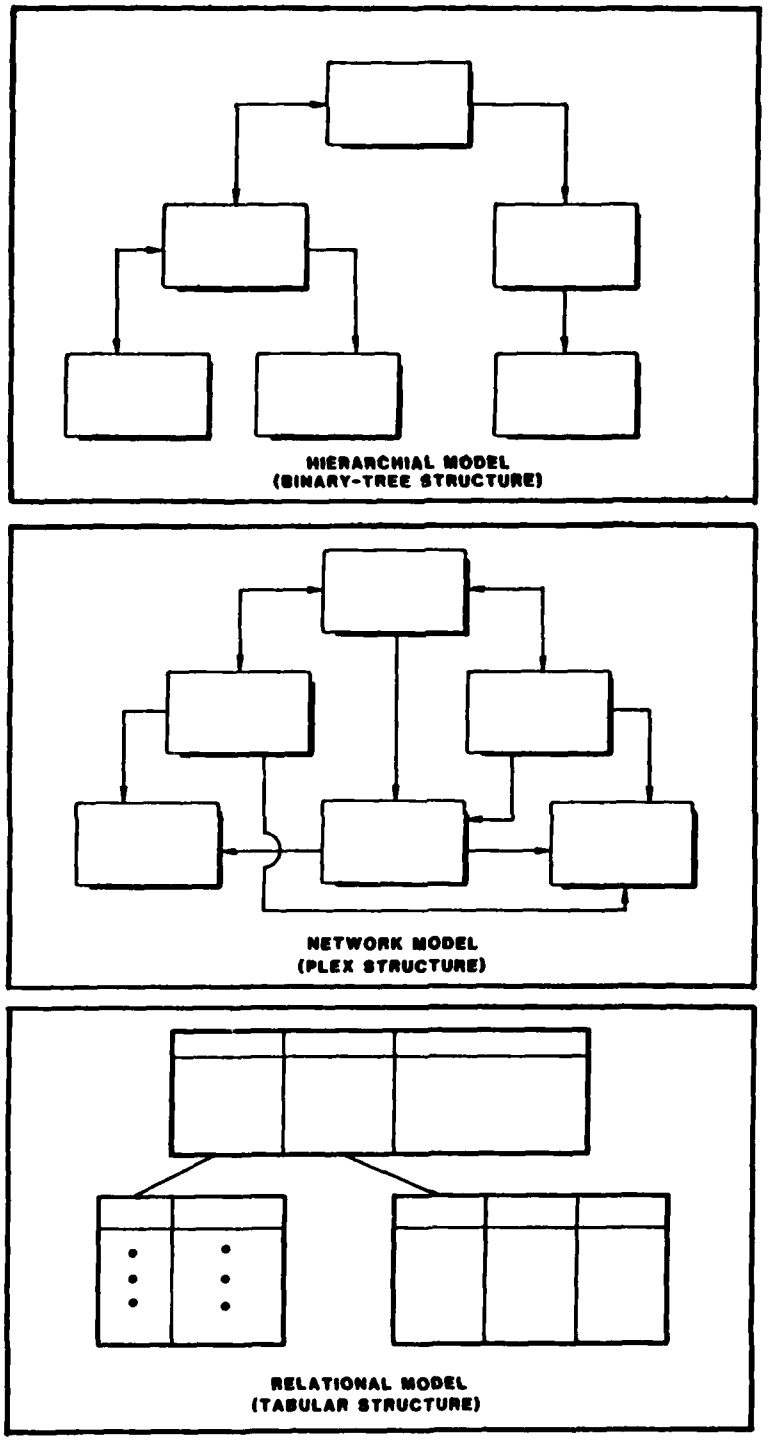


Figure 4-2. Three of the Most Common Models

as relationships of lower-level objects) and generalization (single entity represents a class of individual entities).

Nyerges asserts that the classical relational model loses important semantical information by virtue of its inherent lack of a schema representation of entities and their relationships. Furthermore, relations defining geographic information have two basic problems. Insertion or deletion of tuples is a cumbersome process due to the necessity for sequential searching. Also the duplication and redundancy of data (particularly with polygons) inherent to the relational model leads to inefficient storage utilization.

4.2.2 ENTITIES, ATTRIBUTES, AND RELATIONSHIPS

A useful set of linguistic definitions of cartographic objections was developed by Youngmann (1977). The intent was to formulate a description of a map in terms of the properties of its constituent elements and the relationship among them. These descriptions would be used for answering questions about the map and a formal language would be used to embed the linguistic specifications into the data. The concept of defining a set of rules (i.e., grammar) for the arrangement of map elements in a formal language of maps was based on the inherent consistency of map information and its selection and arrangement.

Youngmann showed the potential for creating useful descriptions of maps by means of hierarchical decomposition of map elements. He introduced map description grammar based on atomic units called primitive objectives which permit new language objects (compound objects) to be constructed. In this hierarchical map language the terminal nodes are geographic phenomena called primitive objects.

A map is ascribed specific meaning when the cartographer assigns values to the attributes of the primitives. The number and type of attributes of a map object is usually a function of map scale. Compositional rules provide for the combination of elements in some hierarchical relationship to produce new elements. These compound objects have a meaning through their own attributes plus the attributes of the primitive objects from which they were formed. The combinations are based on the relationships between the attributes of the constituent objects. Table 4-1 illustrates the set of map primitives and their attributes as identified by Youngmann.

A cartographic object is defined by Nyerges as being the representation of an entity, which may be a thing or an event, that is to be utilized in the modeling of a geographic distribution for cartographic purposes. Cartographic objects are characterized by two categories of spatial properties which are attributes and relationships. An attribute is a

Table 4-1.

Map Description Primitives

PRIMITIVE OBJECTS	ATTRIBUTES
POINT	X,Y,V,L
SYMBOL	Xc, Yc, T, L
STRING	X,Y, ALPHA, ZSIZE, YSIZE, RHO, STYLE
SEGMENT	X_1, Y_1, X_2, Y_2, V, L
VECTOR	X, Y, RHO, D, V, L
ARC	X_1, Y_1, X_2, Y_2, R, V, L
CIRCLE	X_c, Y_c, R, V, L
SECTOR	X_c, Y_c, R, RHO, THETA, V, L
RECTANGLE	X_1, Y_1, X_2, Y_2, V, L
POLYGON	X_1, Y_1, X_2, Y_2, ... X_n, Y_n, V, L

X is X-coordinate
Y is Y-coordinate
V is any set of values (V_1, V_2, . . . V_n)
L is any set of labels (L_1, L_2, . . . L_n)
T is any set of symbol prototypes (t_1, t_2, . . . t_n)
ALPHA is a text string
XSIZE is dimension in X
YSIZE is dimension in Y

RHO is rotation angle
THETA is angle of opening
D is distance
R is radius
STYLE is font type style
X_c is center X-coordinate
Y_c is center Y- coordinate

general descriptive identifier of an entity. When an attribute is given a value it will specifically describe an entity. A relationship names an association which one entity has with another (syntactical or semantical). A relationship can be imposed on objects by stating a relation between them through a compositional rule. Table 4-2 illustrates some of Youngmann's relations.

The term "spatial properties" is used as a general name to describe the spatial attributes and relationships concerning cartographic objects (Nyerges 1980).

Attributes of primitive map objects may be either geometric, graphic, or phenomenological. Geometric attributes structurally describe the form of an object or geometrical relationships between objects. Graphic attributes are locational characteristics of objects. Phenomenological attributes are meaningful non-spatial descriptions of objects. Geometrical and graphic attributes are syntactical descriptions, whereas phenomenological or logical associations between objects that permit compound objects to be described based on the geometric, graphic, or phenomenological attributes of primitive objects.

Phenomenological relationships are functional associations between entities that are mostly semantical in nature and yet are not devoid of syntactical characters.

Youngmann defined a set of rules of composition which are operations that may be performed on map elements based on relationships among the attributes of a specific set of map objects. The rules are employed to generate either a map statement, a graphic, or to analyze the map statement. The generated map statement is a logical graph which may be used to retrieve information about map elements within the statement or to portray the statement graphically. Transformational rules restructure objects into different forms such as creating polygons from line segments. Interpretive rules state some conclusion based on an analysis of the constituent elements. Table 4-3 gives a candidate set of composition rules that could be used with the objects and relations previously defined. These rules permit the modification and transformation of map descriptions to make new maps and to answer questions about the maps.

It is important to understand that map descriptions can be formalized in this manner because the various elements in these line graphics are used in consistent manner. Therefore, the rules of formation and arrangement constitute a syntax of map statements. The set of rules or grammer for the arrangement of map elements in various forms permits the creation of different statements in a map language.

Table 4-2.

Relations

RELATIONSHIP	TYPE
CONNECT	Point objects
CONCATENATE	Linear objects at ends
TOUCH	Linear objects anywhere on segment
CROSS	Linear objects anywhere on segment
PARALLEL	Linear objects one to another
ANGULATE	Linear objects at ends
ATTACH	Areal objects at nodes
ADJACENT	Areal objects along edges
OVERLAP	Areal objects
INSIDE	Points, lines, areas in areas
PROXIMITY	Points to points, lines, areas, lines to lines, areas to areas
ALGEBRAIC	V_1 r V_2 where operator r may be $<$, $\leq$, $\neq$, = , $>$, $\geq$, ϵ
BOOLEAN	Unison, Intersection, or complement of a set of objects

Table 4-3.

Compositional Rules

RULE	MEANING
PROPERTY (O_k, a_{ij}, A_j)	This predicate is TRUE if A_j is an attribute a_{ij} of Object O_k
RELATION (r_i, O_k, O_l)	This predicate is TRUE if the relationship r_i holds between objects O_k and O_l
ADD (a_i, O_k)	The operator adds attributes a_i to description of ojbect O_k
DELETE (a_i, O_k)	The operator deletes attributes a_i to description of object O_k
COMBINE (O_k, O_l	Combine O_k and O_l to form new object provided they have passed a predicated test on a property or relation
RELATE (r_i, O_k, O_l)	Establish relationship r_i between O_k and O_l
UNRELATE (r_i, O_k, O_l)	Remove relationship r_i between O_k and O_l

4.2.3 SYNTACTIC AND SEMANTIC REPRESENTATIONS

Youngmann was concerned with the syntactical description of maps (i.e., the orderly correspondence of graphic elements in a map). Picture processing research indicates that much of the information desired from a picture can be found in the syntactic structure. However, there is a great deal of interest in defining the semantic aspects of maps (i.e., their meaning) as systematically as the syntactical approach. Development of semantical map grammars is much more difficult than the syntactical approach and more research in this area should be pursued.

The requirements for a cartographic data base that is independent of specific applications dictate that both syntactic and semantic descriptions of the data are needed. The user should also be able to access data by name, thereby ensuring that programs can be written that are independent of the data. A self-describing (Weller, Palermo, 1979) data base permits the data to be retrieved by name and the rest of the components of the data are stored in the data base itself.

1. The Syntax (format) of the data

2. The Semantics (meaning) of the data

3. The Access Path to the data.

Self-describing data is a necessary precursor towards achieving independence between programs and data. The data base interface is inherently flexible and application-independent.

The graphic data base organization for a Picture Building System advocated by Weller and Palermo, is structured as a set of named segments, each of which is a collection of named files. These files are composed of an ordered set of uniform records each of which consists of a collection of named fields. The user views the file as a table in which each record is a row and each field is a column in the table. The field description information resides with the data and consists of the following:

1. Field Name

2. Data Type

3. Data Length

4. Semantic Description

Figure 4-3 illustrates the graphic data base organization concept. The data is manipulated by binding program variables to files and selecting of records by matching of field names, values, or record numbers. The data in the data base can be used to create a graphic or be interpreted as a picture.

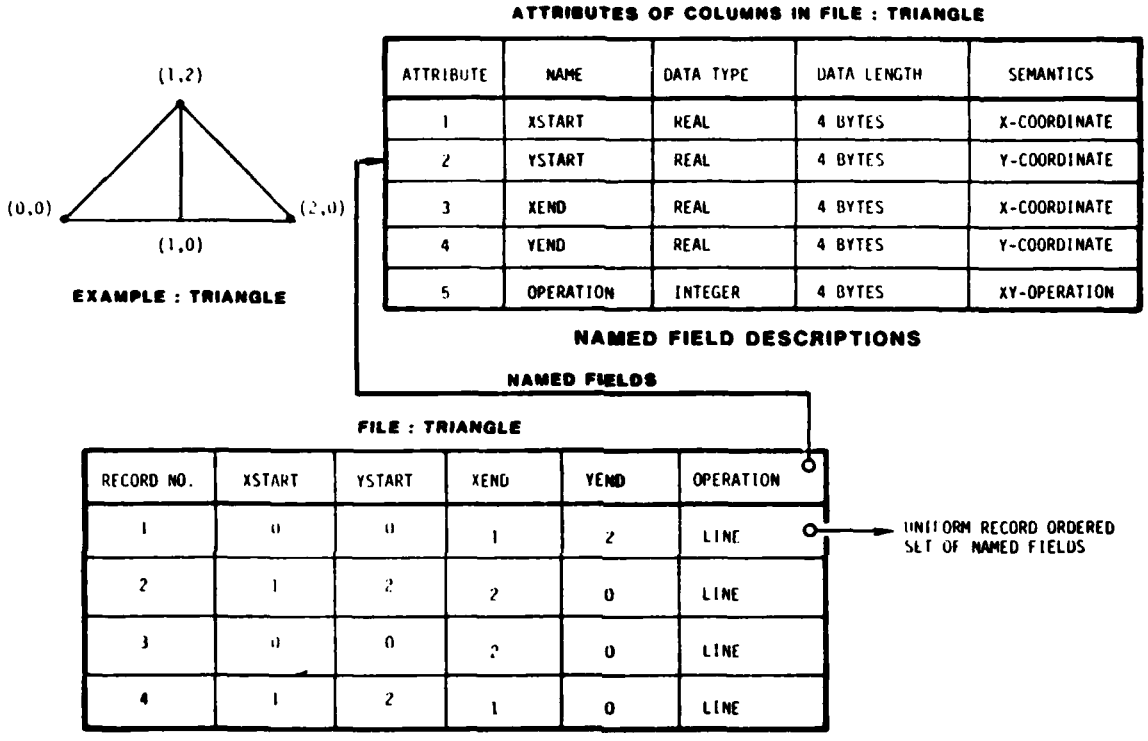


Figure 4-3. Graphic Data Base Organization Concept

In the field of Artificial Intelligence (AI) the nature of knowledge representation has been a major area of research. Data structures and interpretative procedures have been combined to allow intelligence manipulation of the data. The choice of the primitive attributes (facts) which became the foundation of the data base strongly affects the expressive power of the knowledge representation scheme. The degree of control of the interaction between the various data base elements with a design goal towards modularity of data structures are important characteristics of any representation scheme.

One class of knowledge representation formalism which is known as semantic networks, are a useful tool for the notation of entities and relations.

A key feature of semantic nets is that important associations can be explicitly made. The notation consists of nodes (drawn as circles or boxes) and arcs (or links, drawn as arrows) (Weller, Palermo, 1979). Nodes represent entities and arcs represent relations. Relevant facts about an entity can be inferred from the nodes to which they are directly linked. It is possible to establish a property inheritance hierarchy in the net. The semantic interpretation of the net depends on the process that manipulates the net and its capability to infer meaning from the data. Perhaps researchers will one day be able to describe the semantic character of maps by means of semantic nets. Then intelligence processes will be able to interpret the meaning of the map and apply reasoning and inference to the knowledge it represents.

4.2.4 REPRESENTATION THROUGH FRAMES AND SCRIPTS

One of the newest knowledge presentation schemes in AI is dalled frames and scripts. The idea is to present knowledge about events and objects in their most typical or expected form. The frame structure includes declarative and procedural information in predefined internal relations. The frame is to aid in the process of understanding by matching itself to the data it discovers. The script is a frame-like structure designed for representing sequence of events. Together they form a framework for interpreting new data in terms of concept acquired through previous experience. The representational mechanism that makes this reasoning possible is the slot, the place where knowledge fits within the large context created by this frame. The slot permits reasoning based on seeking confirmation of expectations.

Brown (1981) chose a frame-like information structure for a natural language graphic system. The frames represented generic concepts about objects and are represented as networks of nodes and relations which are interconnected to form a data base of facts. The frames are organized in

a generalized hierarchy, with the most general items at the 'top' with the lower levels down a particular branch inheriting the general properties of the top and represented by specific instances of data (slots). This organization eliminates redundancy by means of the property inheritance concept.

Another feature associated with frame structures is that of procedural attachment to slots to drive the problem-solving behavior of the system. Actions attached to frames can be triggered by various situations such as providing appropriate methods to interpret or derive new data; access, delete, or insert data or trigger transfer of control to other frames, trigger procedures or event or data-driven procedures that are highly context and conditionally based.

In Brown's Natural Language Graphic System the information structure consisted of a hierarchy of frame-like information units called chunks, which are either generic or individual. Individual chunks are instantiated forms of generic chunks. Defaults and procedural attachment are used to provide for probable or typical shapes or sizes of objects if they are not provided by the user. The assignment of 'reasonable' values by the defaulting process and the property inheritance hierarchy greatly enhance the apparent intelligence of the system.

Frames and scripts are the most recent attempts by AI researchers to provide a method for organizing large amounts of knowledge to perform cognitive tasks. The ability of these structures to provide direction for active cognitive processing continues to be evaluated and more research is needed. Nevertheless, relative to our raster data structuring research, frames and scripts oriented towards map syntactical and semantical descriptions may offer a potential solution for improving the human interpretability of raster encoded map information.

4.3 Storage Structures

This section discusses the storage structure level of the data organization model presented in Section 4.1 and illustrated in Figure 4-1. The organization of the discussion is based upon "Data Structures for Picture Processing" (Shapiro, 1979), which presented both data structures and algorithms with a focus upon image processing and pattern recognition applications. The recursive/relational data structures presented in this section expand this survey, but employ the same organization and definition of spatial, recursive, graph, hierarchic, and list structures.

4.3.1 RECURSIVE STRUCTURES

A data structure that can be defined in terms of itself is called a recursive structure. In this section we examine some complex recursive structures used in the fields of computer graphics, image processing, and pattern recognition. Two recursive structures that have been investigated in picture processing research are the iconic/symbolic data structure (Tanimoto) and spatial data structures (Haralick and Shapiro).

4.3.1.1 Iconic/Symbolic Data Structure

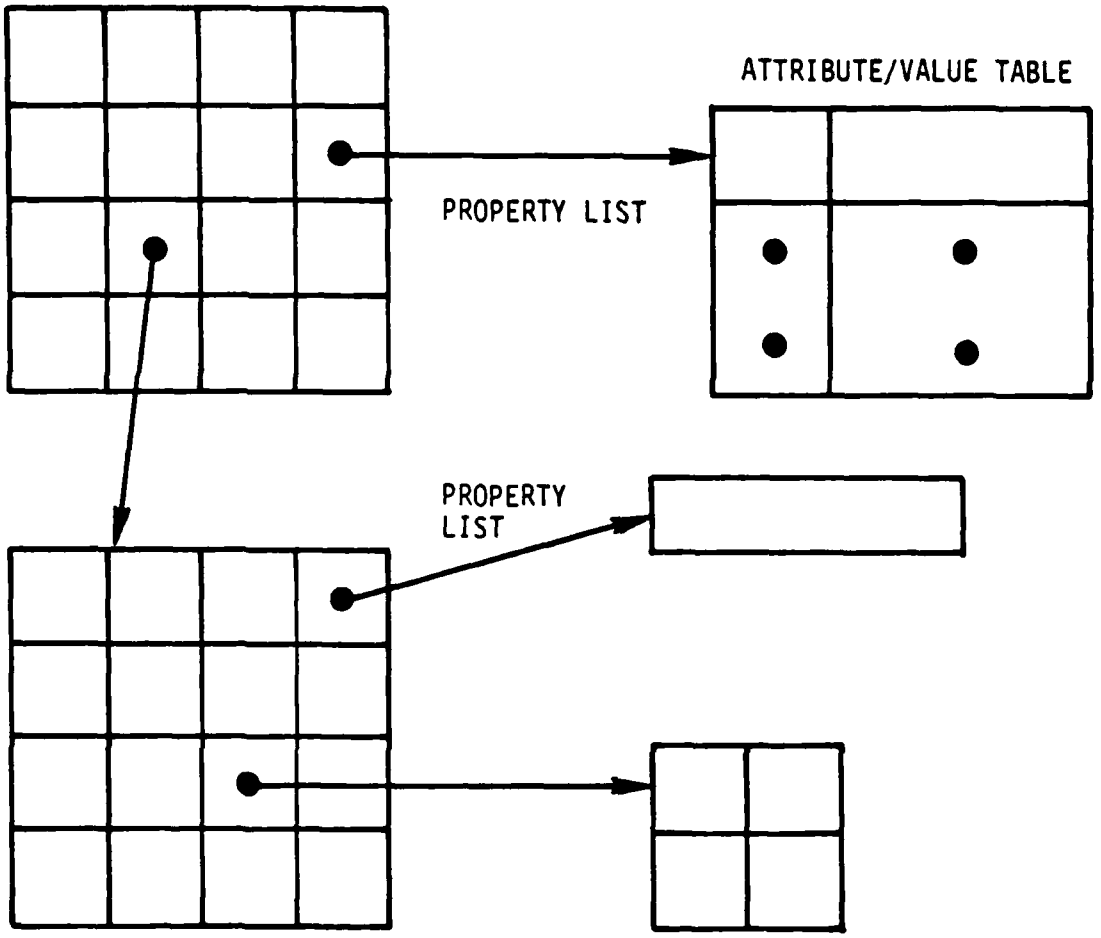
The iconic/symbolic data structure was conceptualized by Tanimoto (1977) for structuring picture data. Iconic information is that which is expressed as digitized visual images. Symbolic information consists of picture properties expressed in relational format. The iconic/symbolic data structure captures the advantages of both representations. Tanimoto's research was concerned with exploring new methods to represent pictorial semantic knowledge in order to improve the image segmentation process.

The distinction between iconic and symbolic is subtle but important to understand in the context of iconic/symbolic data structures. A symbol is an object whose value is associated with it without regard to the physical properties of the symbol itself. A symbol need not share any visual or other similarity with that which it represents. An icon, on the other hand is an image or pattern that bears a natural resemblance to that which it represents (e.g., light intensity as a two-dimensional array).

Tanimoto's iconic/symbolic data structure is a two-dimensional array whose pixels may be intensity values, or pointers to symbolic data structures (represented as relations) or pointers to other icons.

An extended icon is a picture matrix in which the pixels can represent not only the local light intensities (e.g., hue, intensity) but also arbitrary data items. Each pixel has one of its n-bits used as an indicator to flag it as either an iconic or symbolic representation. The remaining n-1 bits provide either the intensity value or the symbol. The symbol may be considered a pointer to some structure. A symbol is an iconic/symbolic data structure, which is a property list or an extended icon. A property list is an array of property values which can be considered a relation. Since a symbol can be a component of an iconic/symbolic data structure, the structure is recursive. Figure 4-4 illustrates the iconic/symbolic data structure. Symbols (pointers) are denoted by arrows emanating from pixels.

The advantages of iconic/symbolic data structures are their efficiency of space, representational power, and facility for query processing.



DEFINITION OF ISDS	
1	ISDS IS A PROPERTY LIST OR AN EXTENDED ICON
2	PROPERTY IS A TABLE OF ATTRIBUTES AND VALUES
3	EXTENDED ICON IS A 2D PIXEL ARRAY
4	PIXEL REPRESENTS AN INTENSITY OR SYMBOL
5	SYMBOL IS AN ISDS

Figure 4-4. Iconic/Symbolic Data Structure

A fundamental objection to the use of pixels both iconically and symbolically in the same icon is the apparent loss of continuity of intensity information where symbols exist. This can be corrected by creating separate arrays for the iconic and symbolic components of the extended icon. Similarly the pixel bit width can be chosen larger than the dynamic range of the intensity values, such that part of the pixel would represent the intensity and the other part would represent the symbol.

With a pictorial data base management system an indexing scheme called "iconic indexing" is used to access information necessary to answer a query utilizing the iconic/symbolic data structure. A "one-level inconic index" is an extended icon whose symbolic pixels point to the data items of the pictorial data base The iconic search mechanism is a procedure capable of answering queries of the general form "Retrieve the information associated with X", where X is one of the following:

1. Absolute X,Y coordinates, the symbol pointer is used to access the desired information.

2. Neighborhood of X,Y.

3. Feature point of a certain type.

4. Grey shade of color value.

There are many potential uses and variations of the iconic/symbolic data structure. Shapiro (1979) used a modified iconic/symbolic structure called a matchform, to preserve the spatial relationships among primitive picture elements, while representing them symbolically. A matchform is simply an array of grid squares, which point to a list of primitive components. Figure 4-5 represents the matchform in which the grid is superimposed over a picture. Tanimoto introduced the notion of iconic symbols which treated a group of pixels, a subimage, as a symbolic entity. The pixels belonging to the group were uniquely identified and a single table of symbol definitions exists for the symbolic entity. Alternatively symbolic pixels may represent procedures that are to be called in the process of examining a locality on a picture. In this way, the knowledge about how different kinds of pictures are analyzed is stored procedurally. This latter use of iconic/symbolic data structures when combined with top-down tree representations (image segmentation) has great applicability in the domain of machine perception.

4.3.1.2 Spatial Data Structures

The spatial data structure introduced by Shapiro and Haralick (1978) is a recursive structure that may be used to represent cartographic objects, their attributes, and multiple relationships among the objects. The

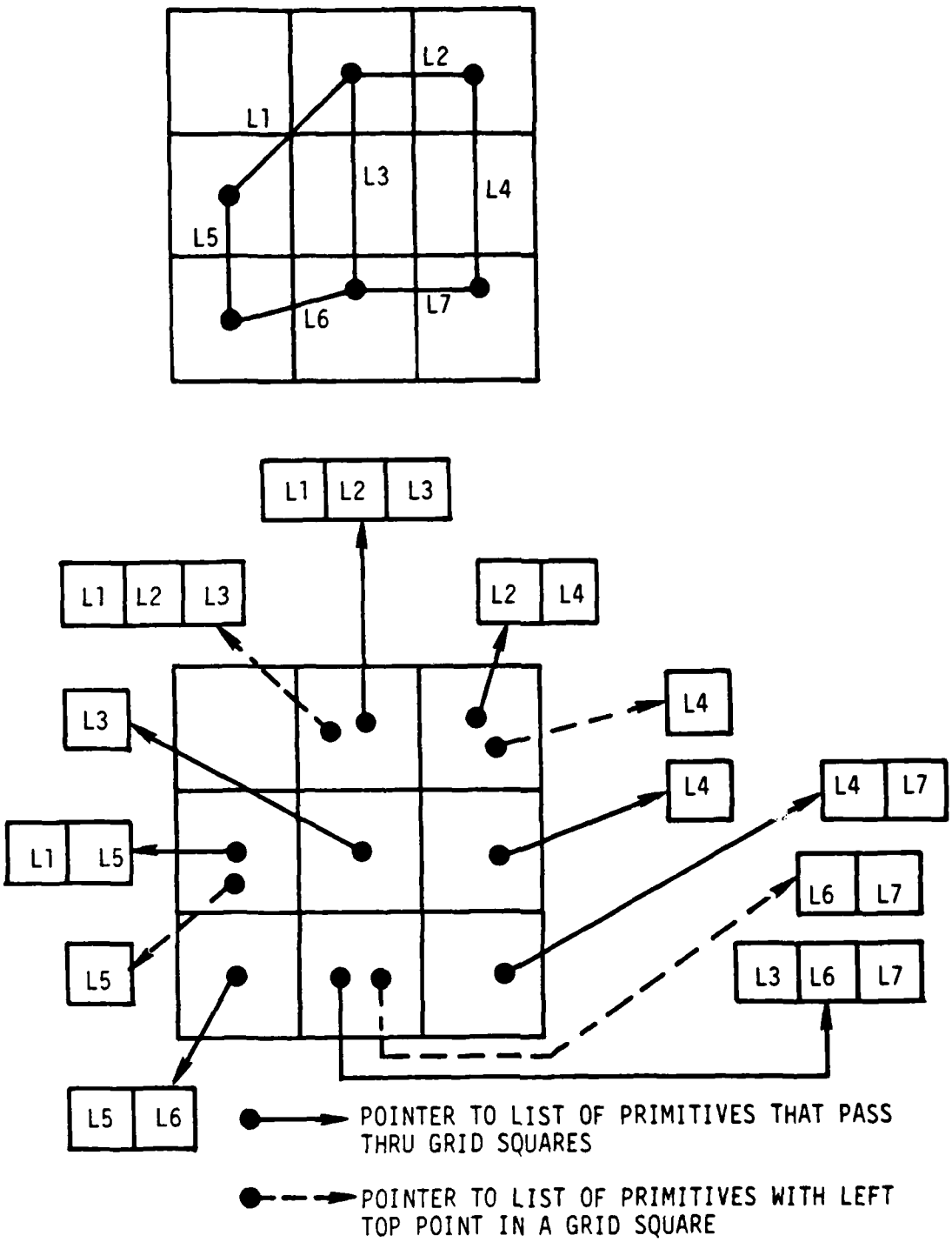


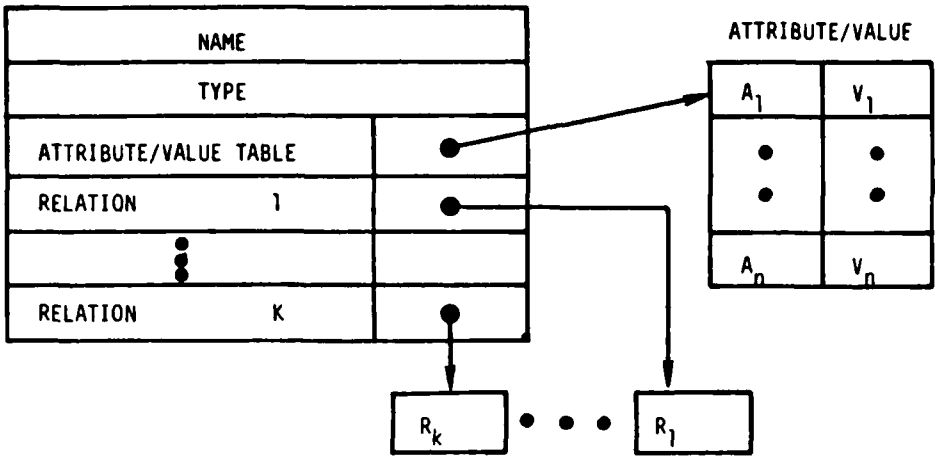
Figure 4-5. Matchform of a Simple Picture

representation of spatial data in a uniform structure permits intelligent interaction in the form of queries and answers. The spatial data structure is sufficiently general, flexible, and efficient to permit storage of any type of spatial information. Figure 4-6 illustrates the logical structure of the spatial data structure. The elements of the spatial data structures may be atoms or spatial data structures.

A node can be represented by the spatial data structure N = (A,Ø), where the attribute/value table A contains information about the node such as its coordinates. A chain is represented by spatial data structures C = (A,R) where A contains length, start node, end node, and R consists of the labeled relation R which ordered the set of points in the chain. A line is represented as L = (A,R) where A contains the attributes of length or number of chains, and the set of relations R contains the unary relation which indicates the linear ordering of the chains that compose the line. A boundary is represented as B = (A,R) where A contains the attributes of the boundary such as the name of the region it surrounds, and R contains the unary relation which is the set of boundaries interior to the outermost boundary. An area can be represented by a spatial data structure whose attributes contain population or similar information and designation for the spatial data structure representing the boundary. The relation set could contain the region adjacency relation, the chain region adjacency relation, and the set of cities or similar information in the area. Likewise the sequence of chains comprising all boundaries of the area would be a labeled unary relation.

A spatial data structure is a recursive structure since both its attribute/value table and its relations may contain spatial data structures. The structure can represent a geographic entity which might be as simple as a point or as complex as a whole map (Haralick and Shapiro, 1979). The spatial data represented in the classical geographic information files such as the U.S. Census Bureau's DIME file can be easily emulated by this spatial data structure (Reference Figure 4-7). Note that the physical storage scheme for the chains is not indicated. The spatial data structure is independent of the physical form of the storage. The spatial data structure can readily handle vector or raster formatted data.

Grid cell data structures explicity represent map areas. The map area entity is a spatial data structure MA = (A_1,P). The attribute/value table A, has attribute THEME with values such as 'soil type' or 'land use'. The partition list relation P consists of a set of ordered (row, partition list) pairs. The 'partition list' is a spatial data structure PL = (A_2,I). The attribute/value table A_2 has the attribute ROW whose value is the number of interval lists composing the partition list PL. The entity 'interval list', IL, is a spatial data structure of form IL = (A_3,A). The attribute/value table A_3 contains the attributes NAME and ROW. The values belonging to NAME are the region names to which the intervals in IL belong.



DEFINITION OF SDS	
1	ATOM IS AN INDIVISIBLE UNIT OF DATA
2	ATTRIBUTE/VALUE TABLE IS A SET OF A/V PAIRS, WHICH MAY BE ATOMS OR MORE COMPLEX STRUCTURES
3	SDS IS AN ORDERED PAIR S = (A,R) WHERE A = ATTRIBUTE/VALUE TABLE AND SET R = $R_1, \ldots, R_k$
4	R_k in R IS AN N_k-ary RELATION ON A SET S_k
5	L_k IS A LABEL SET ASSOCIATED WITH R_k

Figure 4-6. Spatial Data Structure

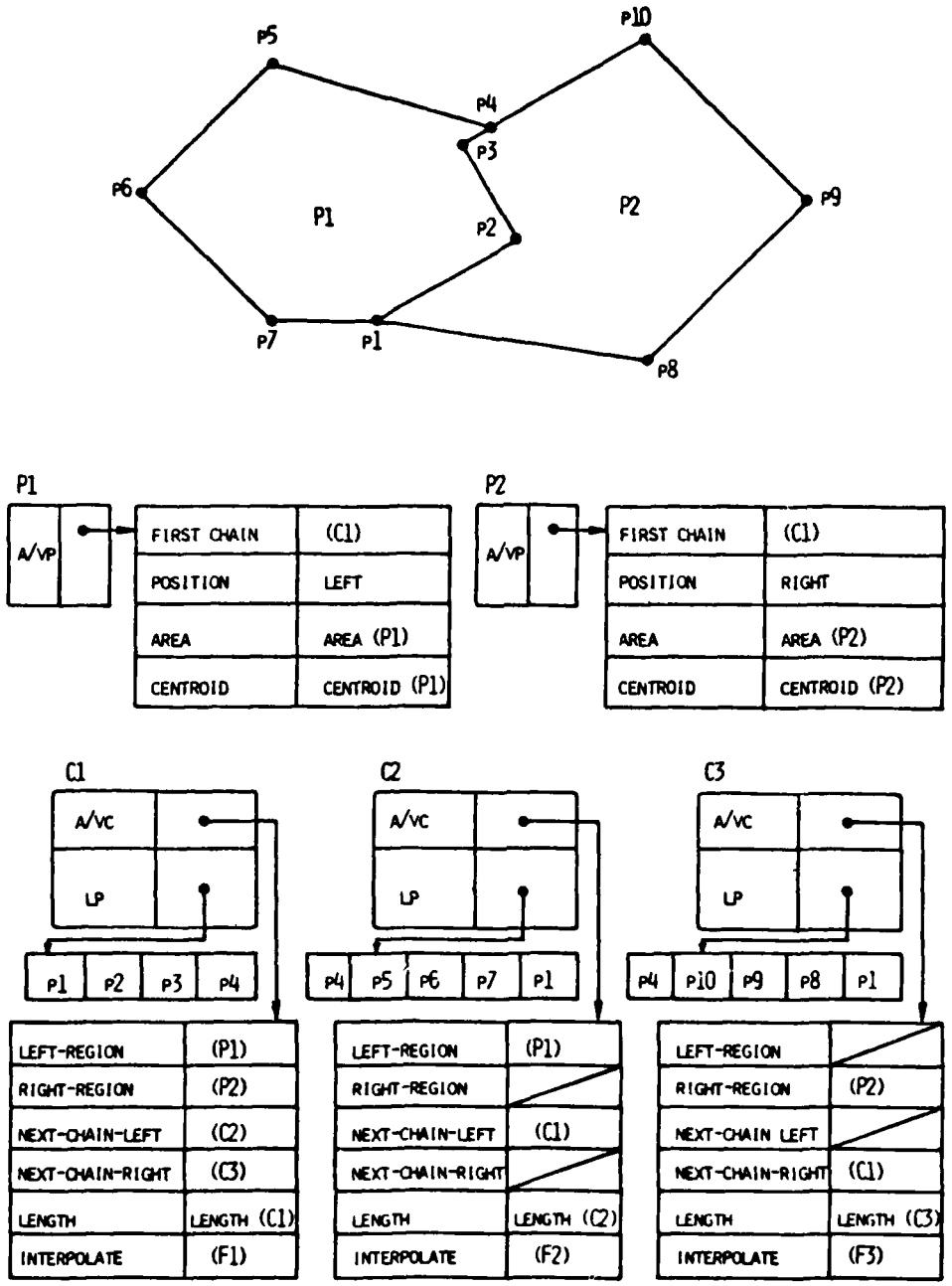


Figure 4-7. Spatial Data Structure Emulating the DIME Representation

The values belonging to the ROW attribute are the row numbers. The H relation is the ordered list of horizontal intervals in the interval list. Each strip in H is an ordered pair of beginning pixel address and ending pixel address with the strip. Figure 4-8 illustrates some spatial data structures as applied to raster grid cell data. The general spatial data structure can easily accommodate either raster or vector formats. The advantage of organizing the data in the general spatial data structure is the ease of handling inferential reasoning by means of an intelligent query capability.

The spatial data structure is a recursive structure since both its attribute/value table and relations may contain spatial data structures. Each spatial data structure has one distinguished property of the entity that the structure represents, which is the attribute/value table. Since the A/V table is a relation the spatial data structure is a set D of relations = $R_1, \ldots, R_k$.

4.3.2 RELATIONAL STRUCTURES

Relational structures have played an important role in the emergence of picture data base technology. A picture description is a relational structure which involves parts of the picture, properties of the parts and relationships among them (Rosenfeld). With the relational structure, it is possible to answer questions about the picture and to formulate statements about it based on the definitions of the parts, properties, and relations. Typically, simple relational structures are composed by segmenting the picture into parts, measuring properties of the parts and establishing relationships between them. Since relational structures permit picture entities to be described at many levels, a hierarchical organization is often utilized. At each level different types of relational structures may be appropriate to adequately describe the picture detail. In this section we discuss three distinct but similar research efforts employing relational structures. Figure 4-9 illustrates the basic concepts of conventional relational structures and relational algebraic operations.

Chang (1977) et. al., designed an integrated data base for pictures and relational tables and provided the means of pictorial information retrieval using a relational query language. The main idea behind Chang' relational data base for pictures is to represent the pictorial information in an iconic/symbolic fashion in the form of logical pictures and physical pictures. The logical picture is a hierarchically structured collection of picture objects which are stored as relational tables in a relational data base and manipulated by means of a relational data base manipulation language. The attributes of these picture objects are also handled by the relational data base management system.

Haralick and Shapiro (1982) et. al. have implemented an experimental relational data base system for cartographic applicatio.., which is based

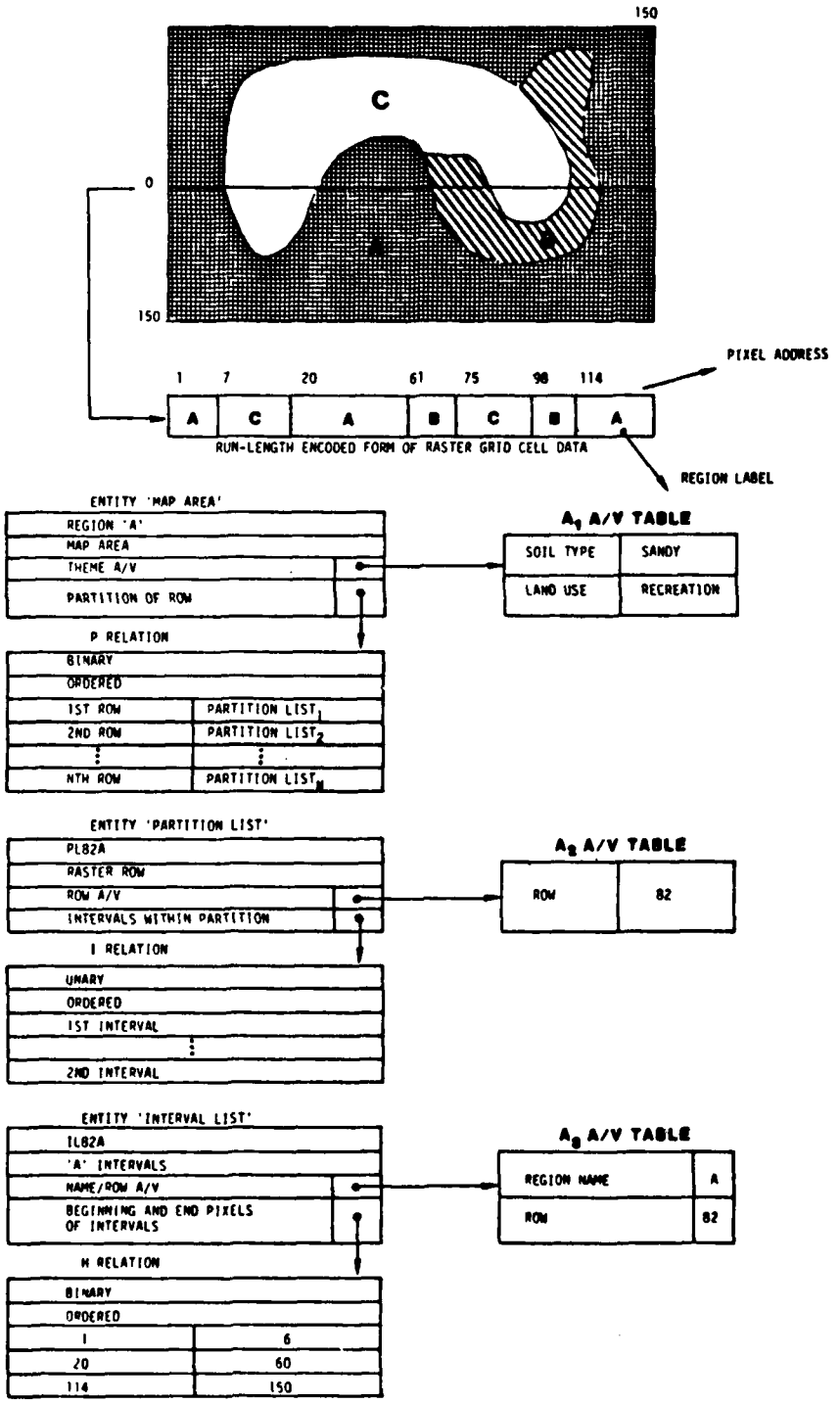
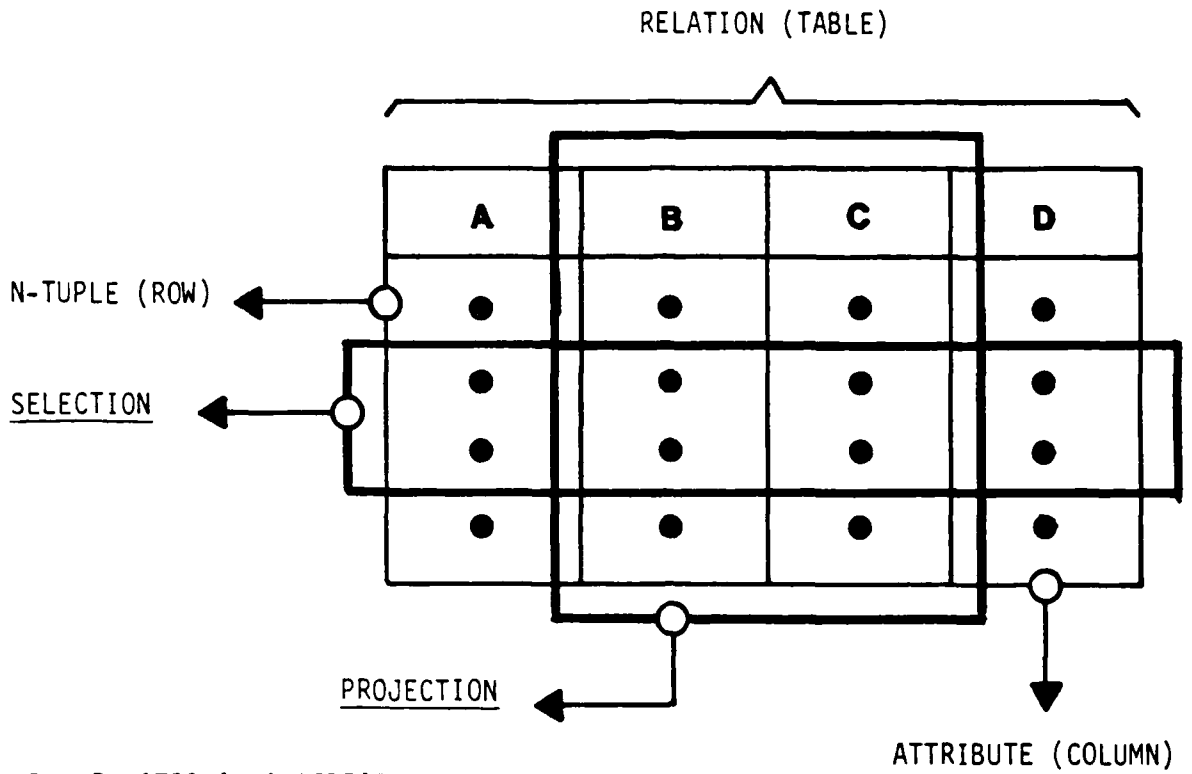


Figure 4-8. Spatial Data Structures Applied to Raster Data



NB. RELATIONAL ALGEBRAIC
OPERATIONS ARE UNDERLINED

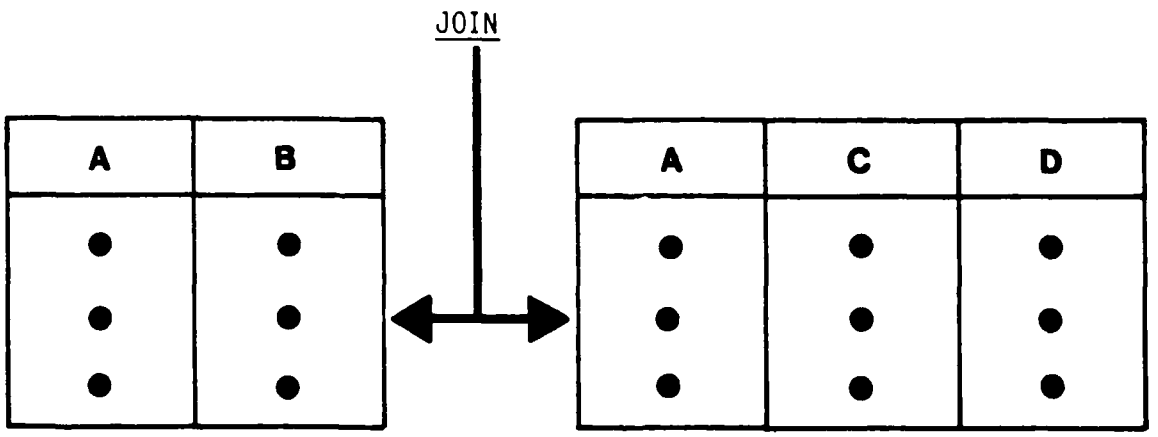


Figure 4-9. Basic Concepts of Conventional Relational Structures and Relational Algebra Operations

on a relational approach for designing spatial information systems. A relational query language for interacting the geographic entities and relationships is described.

Nyerges (1980) and Chen (1976) have performed research involving a particular spatial data structure known as the entity-relationship model which is very similar in nature to Shapiro and Haralick's spatial data structure concept.

All threee of these works are relevant to the IRDS effort because they attempt to solve the problems associated with the structured data organizations which make them unsuitable for cartographic/geographic query processing.

4.3.2.1 Entity-Relationship Model

The entity-relationship model was originally conceptualized as an information structuring method for representing real world entities. The basis for the entity-relationship model was the desire to unify the network and relational models with the entity set model (Chen, 1976 and Nyerges, 1980).

In this document we have defined entity, relationships and attributes (Section 5). We summarize our previous discussion based mostly on the work of Nyerges and Youngmann.

1. Entity is a cartographic object.

2. Entities (e_i) are classified by membership within entity sets (E_j).

3. For each entity set (E_j) there is a predicate which is used to test whether an entity (e_i) is a member.

4. An entity e_i that belongs to entity set E_j, which in turn belongs to entity set E_n also belongs to E_n.

5. Relationship is a graphical, geometrical or phenomenological asscciation between entities.

6. Relationships are classified by membership within relationship sets.

7. A relationship set is a mathematical relation defined in members of entity sets.

Each n-tuple $[e_1, e_2, \ldots e_n]$ comprised of elements $e, \varepsilon E_1, e_2, \varepsilon E_2, \ldots \varepsilon e_n$ represents a relationship (r_i) belonging to relationship R_i. Each entity performs a function in the relationship.

8. For each relationship set R_i, there is a predicate to test the membership of $r_i \varepsilon R_i$.

9. Information about an entity or relationship is expressed by a set of attribute/value pairs.

10. Values are classified into value sets and there is a predicate to test whether a value belongs to a particular value set.

11. An attribute is a function which maps from an entity set or a relationship set into a value set or cartesian product of value sets.

$$f: E_i \text{ or } R_i \rightarrow V_i \text{ or } V_{i_1} \times V_{i_2} \times \ldots \times V_{i_n}$$

4.3.2.1.1 Entity-relationship diagram. The information structure of relations can be represented by a network diagram. A tabular form of relations only represents the logical data structure. The entity-relationship diagram provides a representation scheme that clearly portrays information concerning entities and relationships. There are five different structural representations for the various type of relationships among entities (reference Figure 4-10). An entity-relationship diagram is comprised of these structural representations. All cartographic relationships between objects as identified by Youngmann can be represented within this framework. The difference between the aggregation of entity sets to produce relationship sets versus compound entity sets is that with the former the entities remain autonomous; whereas with the latter the entities lose their autonomy to a higher level (compound) entity. The capability to distinguish between these two types of aggregation is crucial for representing entities and relationships in a spatial data base. In Figure 4-11, an example of entity-relationship diagram for a cartographic data base is presented. The links in this diagram are provided by primary keys which allows bidirectional flow. The intent is to provide an unambiguous definition of the information structure of the cartographic data base.

4.3.2.1.2 Data structure for entity/relationship relations. The entities and relationships of the information structure can be represented in a data structure. An entity is represented by an entity key, which is one or more attributes that are a unique mapping from the entity set to a group

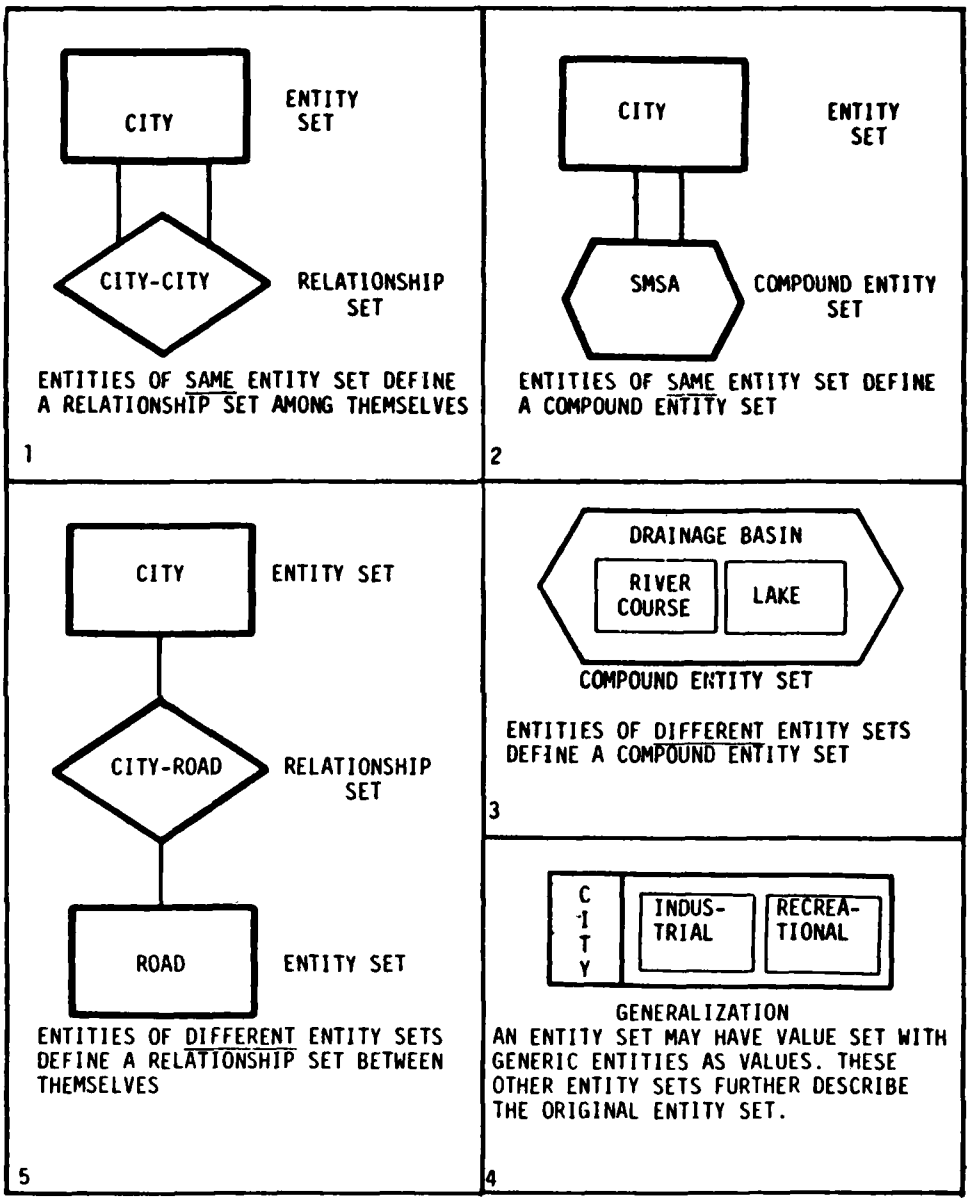


Figure 4-10. Structural Representations of Relationships Among Entities

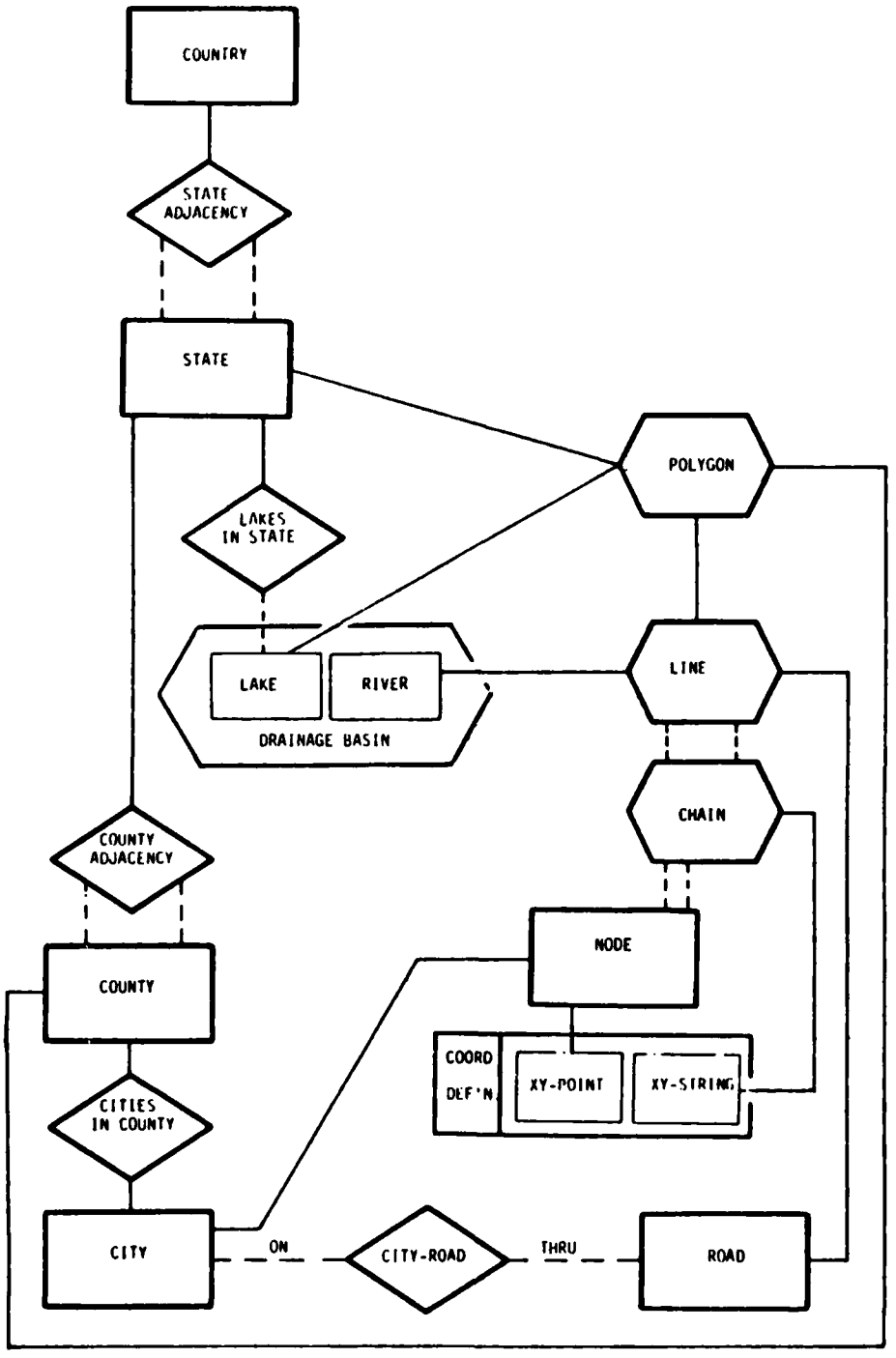


Figure 4-11. Entity-Relationship Diagram Examples

of value sets. When multiple entity keys exist a primary key is used for primary identification. Values for primary keys in a relation are the links between entities. Information about entities in an entity set are organized in an entity relation (Figure 4-12).

Information about relationships can also be organized in a relationship relation (see Figure 4-12). The entities involved in this relationship are represented by their primary key. The primary key in the relationship relation is a concatenated key comprised of these primary keys. The primary keys are the links to their respective entity sets (i.e., no pointers are necessary). Additional characteristics of the relationship relation are the entity relation names, role of each entity or function that the entity performs in the relationship, and the entity attributes.

The logical data structure of the entity-relationship model utilizes the tabular form of the relational model. The chief advantage of this data structure is its simplicity by virtue of the use of keys rather than pointers. The inclusion of relation names in other relations serve to act as keys to completely avoid the use of a complex pointer scheme. The entity-relationship model permits the direction association of relationship relations with entity-relations which is a highly desired property of a spatial data structure.

4.3.2.2 Entity-Oriented Relational Data Base

An experimental entity-oriented relational data base system for cartographic applications has been implemented at Virginia Polytechnic Institute (see Vaidya, Shapiro, Haralick, Minden, 1982). The relational approach to designing a spatial information system permits either raster or vector data or both to be managed in an integrated manner suitable for user query. The keystone of this system is the general spatial data structure (see Section 5.4.1.2).

The first step in the implementation of the data base is the design of the data base's conceptual model (data model). The conceptual model is the representation of the complete information content of the data base at a fairly abstract level of detail. The description of the model at the level detail that permits the data base to be implemented is called the schema. The schema is developed in the form of a prototype structure for each type of spatial data structure. The prototype indicates the attributes that are to be in the attribute/value relation and what the other relations are that will comprise the data structure. The spatial data structure prototypes for the regions, water streams, stream, label, polygon, and chain entities and their respective relations are illustrated in Figure 4-13. The spatial data

PRIMARY KEY

CITY-ID	NAME	POSITION		POPULATION	← ATTRIBUTE
CITY-ID	NAME	LATITUDE	LONGITUDE	NO. OF PERSONS	← VALUE SET
1	ROME	43° 7'	75° 38'	35,000	← ENTITY TUPLE
•	•	•	•	•	
•	•	•	•	•	

ENTITY-RELATION EXAMPLE

CITY	ROAD		← ENTITY RELATION NAME
ON	THRU		← ROLE
CITY-ID	ROUTE NO.	TRAFFIC VOL.	← ENTITY ATTRIBUTE
CITY-ID	ROUTE NO.	NO. OF CARS	← VALUE SET
1	26	14,000	← RELATIONSHIP TUPLE
•	•	•	
•	•	•	
•	•	•	

RELATIONSHIP-RELATION EXAMPLE

Figure 4-12. Examples of Entity-Relation and Relationship-Relation

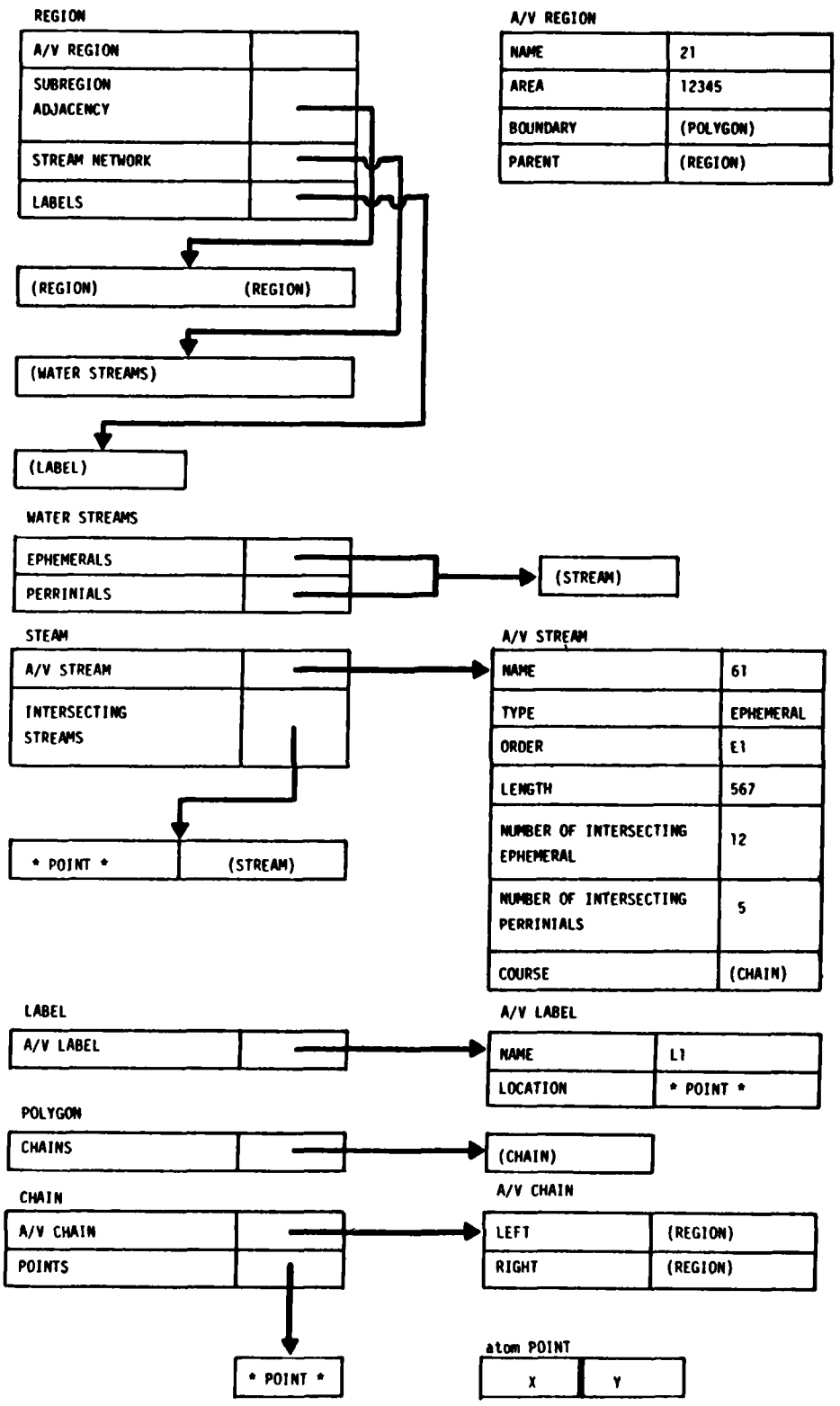


Figure 4-13. Spatial Data Structure Prototypes

structure prototypes provide the system with a knowledge base for understanding the spatial relationships among the entities. The prototypes indicate the access paths through the data structures in order that answers to queries may be fulfilled.

Vaidya, et. al. asserts that a conventional relational query language is insufficient to provide the types of answers to queries they envision for their system. Essentially the objective of the VPI experimental system is to prove the feasibility of developing an intelligent system with natural language query facilities. The system would use the knowledge of the spatial data structure prototypes and semantics of the various tuple components to invoke deductions that could determine the best sequence of relational operations required to extract this information from the data base.

Currently, the experimental development has demonstrated that geographic data can be represented in the entity-oriented relational framework. A stack-oriented query language is provided for the user-data base interface. Users can define constants, variables, and arrays and perform primitive operations using control structures. Standard arithmetic and relational operators are provided. Control primitives include two IF-THEN-ELSE, DO WHILE constructs. The future intelligent system with natural language query capabilities would permit requests of the form:

```
FIND ALL RIVERS
     WITHIN N MILES OF CITY A,
     LONGER THAN M MILES,
     AND CROSSED BY HIGHWAY B.
```

The stack-oriented query is an initial step towards user-system communication. Much more effort in the research and development of the natural language query language is required.

4.3.2.3 Relational Picture Data Bases

An experimental system developed by the Univesity of Illinois (Chang, 1977) for medical information management research utilizes an integrated data base for pictures and relational tables. The GRAIN (Graphics-oriented Relational Algebraic Interpreter) system combines a relational data base management system with an image storage and retrieval system (reference Figure 4-14). The relational data base contains logical pictures which are a model of the real image. Since the logical pictures contain the structured definition of picture objects, their attributes and sketches of the logical picture as line drawings, it is necessary only to manipulate the physical pictures for most applications. The physical pictures are raster images for which an efficient picture paging scheme has been developed. The relational data storage and image storage are maintained on different devices.

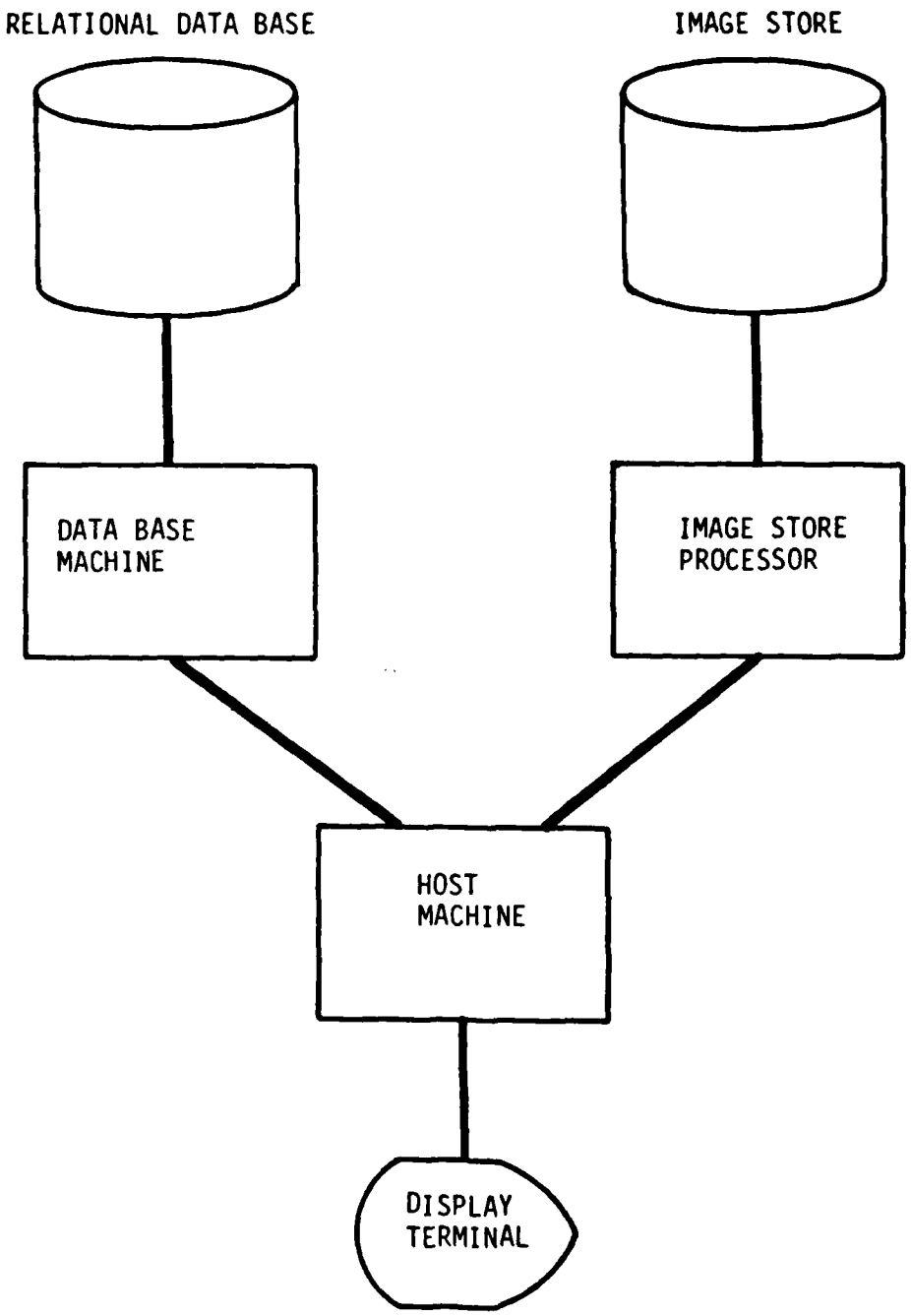


Figure 4-14. Integrated Data Base Concept of Graphics-Oriented Relational Algebraic Interpreter Systems

A logical picture is defined as a collection of picture objects by means of three relational tables: the picture object table (POT), the picture contour table (PCT), and the picture page table (PPT). The relational algebraic commands may be applied to these tables to manipulate picture objects through creation, modification or deletion. (Reference Figure 4-15.) With the POT and PCT tables a picture depicting the contours and labels of picture objects can be generated. A picture object can be retrieved from POT by its logical attributes. A relational algebraic language is used for logical picture object retrieval. The picture pages define the minimum enclosing rectangle for the picture object. The PPT specifies which picture pages are to be accessed from the image store corresponding to the given logical picture object. The contour picture is a line drawing of a logical picture object in coordinates relative to the minimum enclosing rectangle surrounding the object. The digitized picture is the raster image whose coordinates are chosen relative to the corner picture object's minimum enclosing rectangle.

The system permits the retrieval of picture objects by manual selection via a plasma display touch panel or by means of a query language. The user may perform various pictorial information retrievals by zooming, panning, and using spatial relational analysis operations to manipulate the pictorial data base.

A pictorial data base interface called GRAIN is an integral feature of the system (LIN, 1980). GRAIN is a picture query translation system which translates user queries into picture algebra operations according to syntax rules and a pictorial data base skeleton. The pictorial data base skeleton includes a pictorial relational schema and a pictorial conceptual schema. The data base skeleton is a condensed description of a data base which reflects the user's view of the data and the intended usage of the data base (information structure). It contains functional relations among data base entities, semantic relations among data base entities and contents description of data base entities, which includes relational files and attribute descriptors. The concept of data base skeleton is extended to model the pictorial data base which includes spatial relations, structural relations and similarity relations among pictorial data base entities.

The functional relations include the dependency relation, full dependency relation, equivalence relation, containment relation, and attribute similarity relation. The spatial relations include the binary relations of on, adjacency, through, and intersection. A special relation between attribute descriptors and geometrical description is provided to handle user query's that involve both a verbal description or geometrical description of a desired picture object. For example, the area size of and entity may or may not be stored as an attribute relational file when this type of query cannot be satisfied by the attribute relation file the

NAME	OWNER NAME	LABEL POSITION		ANGLE	ATTRIBUTES			
		X	Y		A_1	A_2	A_3	A_4
A	D	i	j	o	1	o	o	o
●	●	●	●	●	●	●	●	●
●	●	●	●	●	●	●	●	●
●	●	●	●	●	●	●	●	●

PICTURE OBJECT TABLE

NAME - OBJECT'S UNIQUE NAME

OWNER NAME - NAME OF PICTURE OBJECT HAVING THIS OBJECT AS A COMPONENT

LABEL POSITION - OPTIONAL LABELING

ANGLE - OBJECT'S ORIENTATION RELATIVE TO OWNERS REFERENCE SYSTEM

ATTRIBUTES - CHARACTERISTICS OF OBJECT

NAME	TYPE	CONTOUR FILE NAME	CONTOUR CODES
A	4	PICA	$(x_1, y_1), (x_2, y_2)$ $(x_2, y_2), (x_3, y_3)$ $(x_3, y_3), (x_4, y_4)$ $(x_4, y_4), (x_5, y_5)$
●	●	●	●
●	●	●	●

PICTURE CONTOUR TABLE

NAME - OBJECT NAME

TYPE - INDICATES CONTOUR DATA FORMAT

CONTOUR FILE NAME - OPERATING SYSTEM FILE NAME OF CONTOUR FILE

CONTOUR CODES - DEPICT GRAPHIC OUTLINE

NAME	NPAGE	LOCATION		ANGLE	PAGE SIZE	
		x_p	y_p		l_x	l_y
A	2	D.	4	0	2	2
●	●	●	●	●	●	●
●	●	●	●	●	●	●
●	●	●	●	●	●	●

PICTURE PAGE TABLE

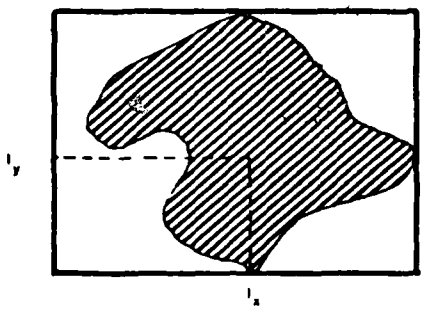
NAME - OBJECT NAME

NPAGE - NUMBER OF PAGES BELONGING TO PICTURE OBJECT

LOCATE - LOCATION OF LOWER LEFT CORNER OF PAGE

ANGLE - PICTURE OBJECT ORIENTATION

PAGE SIZE - PAGE HEIGHT AND WIDTH



COVERING OF MINIMUM EXPLOSIVE RECTANGLE BY RECTANGULAR PAGES

Figure 4-15. Logical Picture Representation by Relational Tablets

pictorial query translator must execute a picture operation to compute "area" by first locating the geometrical relational file for the entity of interest. This capability supports spatial analysis and similarity computations. If spatial relational files, similarity index relation files or attribute descriptors do not exist, the pictorial query translator utilizes an embedded conceptual graph representing the pictorial conceptual schema.

Figure 4-16 illustrates the four types of retrievals provided by the natural GRAIN query language. GRAIN was developed because traditional relation algebraic query language was inadequate for analysis of spatial relationships among picture objects.

The Query-by-Pictorial Example (QPE) relational query language, which was developed by Chain and Fu (1980), of Purdue University, is another prominent pictorial query language. The QPE language was implemented within the Integrated Image Analysis and Image Data Base Management System (IMAID), which is an image processing and image recognition system. By using image processing and pattern recognition manipulation functions, structures and features (picture objects) are extracted from images. These extracted descriptions are integrated into the relational data base, while the 20 raster images are stored separately in an image storage device.

In summary, the classical relational model must be generalized to accommodate pictorial data and its various levels of abstraction. Many proposed extensions have been reported and future research should lead to the development of more sophisticated pictorial relational models. Their utility in practical application domains will undoubtedly depend upon the flexibility of the user interface. Hopefully new research advances will facilitate the development of pictorial information systems for improved understanding and general problem solving.

4.3.3 GRAPH STRUCTURES

General graph structures representing relations between elements have been used for shape matching (pattern recognition), segmentation, and boundary tracing in picture processing. Preceding the presentation of specific graph structures is a discussion of the basic notions of graph theory. This provides the background for the subsequent discussion of Region Adjacency Graphs, Line Adjacency Graphs, and Hypergraphs.

4.3.3.1 Graph Theory

Graphs have been used in applied mathematics and other sciences since Euler (1707-1782) used graphs to settle the Konigsberg Bridge Problem. (Euler, 1736). Graphs provided a diagrammetric description of abstract

RETRIEVAL TYPE	RETRIEVAL KEYS	QUERY COMMAND EXAMPLES
ATTRIBUTE RETRIEVAL	LOGICAL ATTRIBUTES IN RELATIONAL FILES OR CONCEPTUAL GRAPH	GET POPULATION, AREA; PICTURE OBJECT EQUALS CITY ABC
SPATIAL/STRUCTURAL RETRIEVAL	SPATIAL RELATIONS OR STRUCTURAL PROPERTIES	SKETCH HIGHWAY; THROUGH CITY NAME EQUALS CITY ABC AND CITY XYZ
SIMILARITY RETRIEVAL	SIMILARITY RELATIONS	SKETCH HIGHWAY; SIMILAR TO HIGHWAY NAME EQUALS 1365 USING R1 (R1 IS SIMILARITY RELATIONAL TABLE
COMPLEX RETRIEVAL	COMBINATIONS OF ALL THE ABOVE	SKETCH CITY; WEST OF CITY ABC AND AREA GREATER THAN N.

Figure 4-16. GRAIN Picture Query Retrievals

problems which parallel the human intuitive processes. The nodes (points, vertices, simplexes, branch nodes, elements) represent elements (facts) whose relationships to other nodes is expressed by the connecting arcs (edges, lines). The nodes and relations may have values or weights depending on the application, but basically they simply connect the nodes by some criterion. The advantage is the graphic representation of a relational concept presented for human understanding. A science of mathematical theory (Graph theory) is developed to supplement the pictoral representations and express the relational properties of the diagrams. We will discuss some of the practical roots of the science and the application to image processing in the next few paragraphs. This discussion will avoid the associated mathematics and concentrate on the diagrammetric properties.

For a starter, we return to the Konigsberg Bridge of Euler which illustrates the geometrical utility of the theory. The problem was of a trivial nature to the world, but its solution has had tremendous impact. In Figure 4-17, the park in Konigsberg with its seven bridges linking two islands in the Pregel River is depicted. The problem was to visit all land areas crossing each bridge exactly once. Euler showed that a node (land area) must be incident (connected) by an even number of arcs (bridges) to be traversed only once if each node is to be visited only once.

The next major application of Graph theory is the Kirchoff (Kirchoff, 1847) solution to electrical circuits. Each circuit is replaced by its graph and solved simultaneously by the principal of superposition. Cayley used graph theory to represent the familiar chemical relationship of atoms in the organic molecules of chemical isomers and discovered the class of graphs termed trees (Cayley, 1881).

Graphs have since become a common technique for most sciences from psychology to computer science. The application with which we are most concerned is the raster image. There are primarily two areas with which we will be concerned - the pyramidal structure and the unrestricted graph. Graphs are a general term for graphs, undirected graphs, multigraphs, directed graphs, digraphs, subgraphs, pseudographs, etc. For uniformity, the terms used here are node and arc along with several of the graphs - multigraph, directed graph, and subgraph. Figure 4-18 illustrates planer graphs which include:

1. An undirected graph with four nodes, five arcs, and three faces (the two tirangular areas and the exterior area).

2. A multigraph can have more than one arc between two nodes.

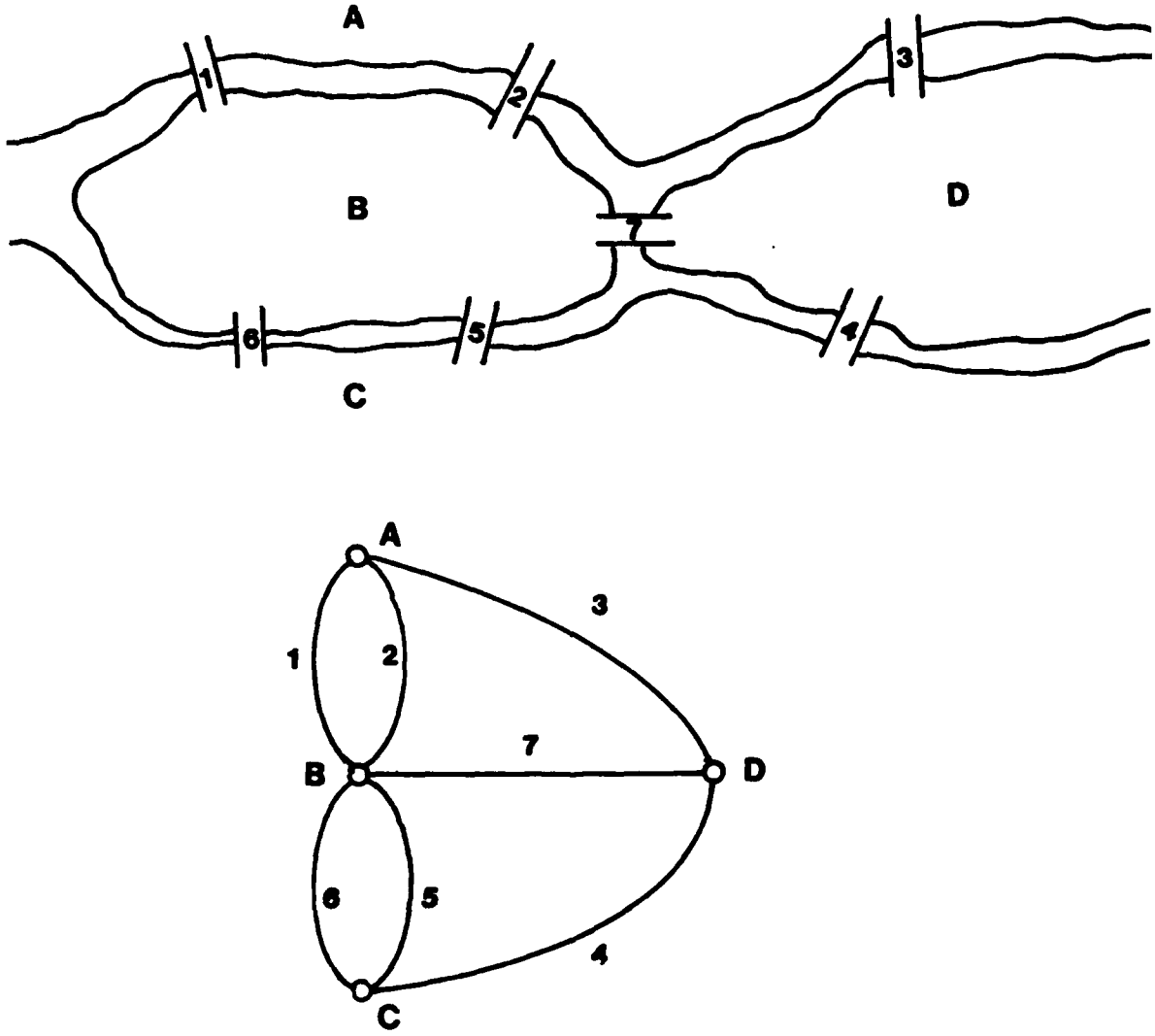
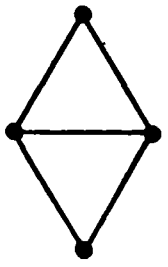
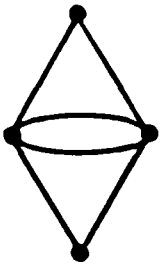


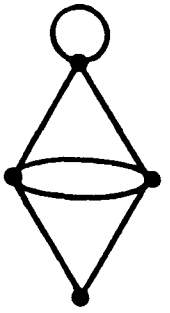
Figure 4-17. Konigsberg Bridge Problem



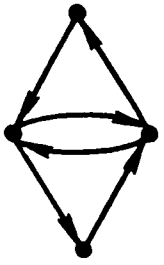
a. GRAPH OR UNDIRECTED GRAPH



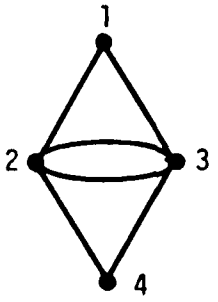
b. MULTIGRAPH



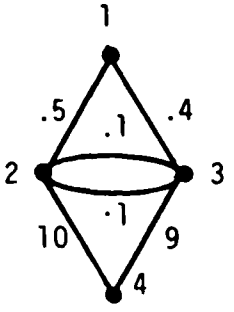
c. PSEUDOGRAPH



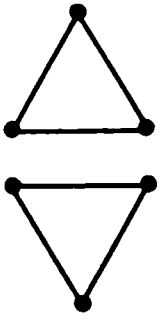
d. DIRECTED GRAPH (DIGRAPH)



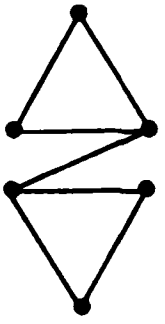
e. LABELED GRAPH



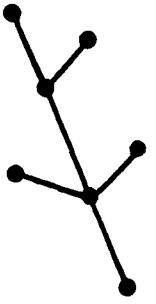
f. WEIGHTED GRAPH



g. DISCONNECTED GRAPH (2 SUBGRAPHS)



h. GRAPH WITH BRIDGE



i. TREE

Figure 4-18. Planar Graphs

3. The pseudograph allows arcs to begin and terminate at the same node.

4. A directed graph (digraph) has direction associated with every arc.

5. A labelled graph identifies the nodes and arcs, i.e., node 1, node 2, arc (1,20).

6. Weighted graph assigns weights to the relation of one node to another.

7. A disconnected graph has no arcs between subgraphs.

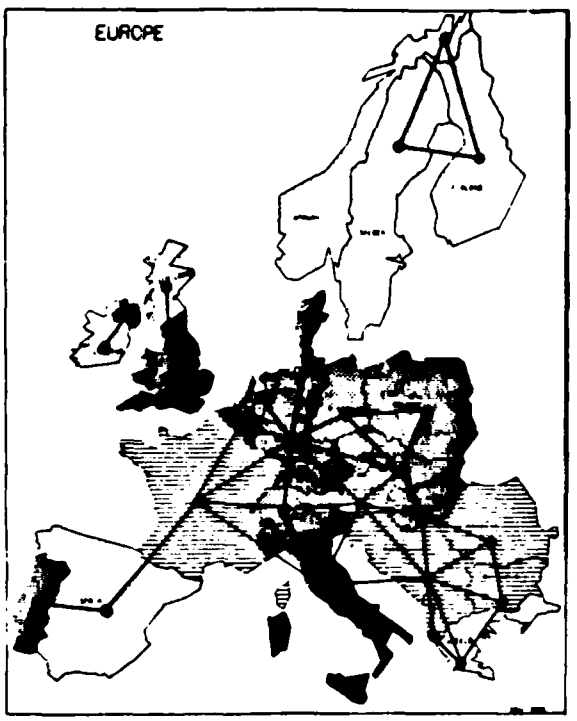
8. A bridge is an arc that when removed creates a disconnected graph.

9. Trees are a special case of graphs which will be dealt with in subsequent sections.

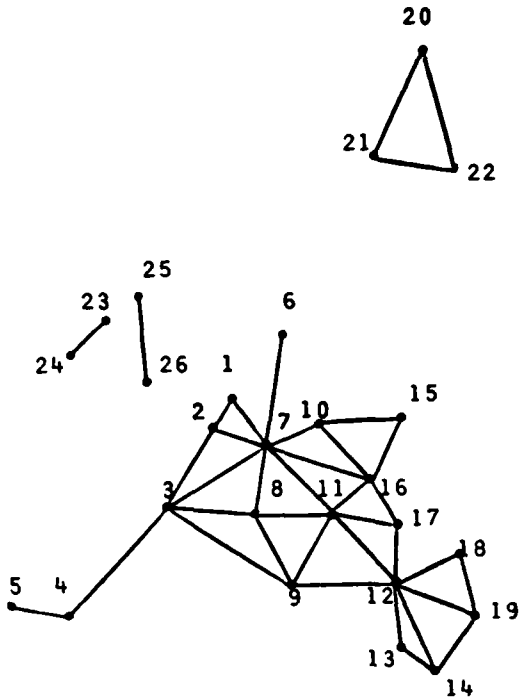
Graph theories have been applied to picture processing by considering each pixel as a node and defining relations between pixels by arcs. The two most important areas are graph structures and hierarchic structures. Both are concerned with graphs but differ in the structures used. If we describe a region within a picture, the hierarchical structure will describe the region by the pixels within the region. A graph structure may show the boundary of the region or its relation to neighboring regions.

4.3.3.2 Region Adjacency Graphs (RAG)

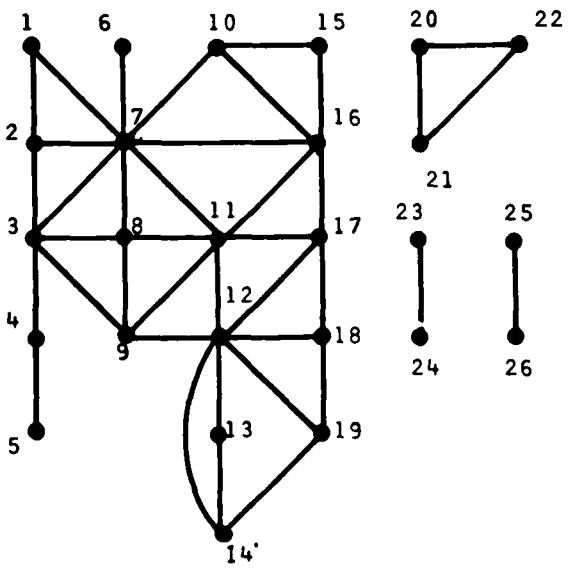
Region Adjacency Graphs are the classical approach of applying graphs to mapping and charting problems. Airline route maps are a good example where the nodes represent cities serviced and the arcs are the connecting routes. Figure 4-19a shows a RAG of Europe where each country is represented by a node and its land neighbors are represented by the arcs. Figure 4-19b shows the RAG without countries. Figure 4-19c is an equivalent RAG since the lengths of arcs and positioning of nodes have no significance to the RAG. Another way of representing the same RAG is the adjacency matrix. Taking Figure 4-19d and addressing each node as a row and a column; a 1 is entered in the matrix whenever an arc connects a node to another. This forms a sparse diagonal matrix representing the Region Adjacency Graph. Since this example is of a bidirectional graph, the diagonal property of the matrix allows storing half of the matrix. Values can replace the 1's of the matrix indicating such things as language similarity, political compatibility, ease of access or military vulnerability.



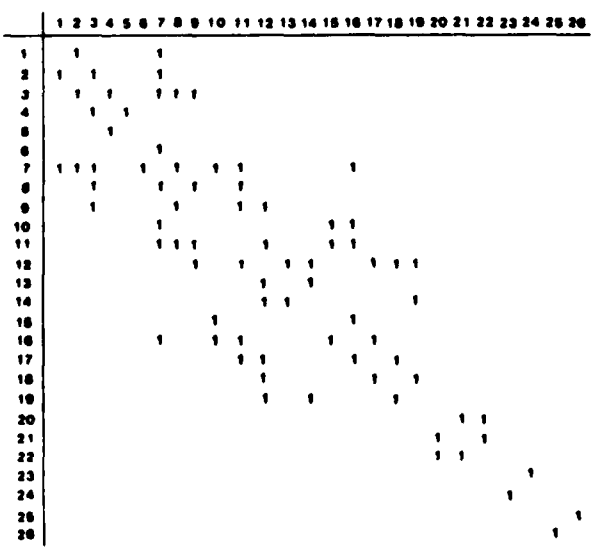
a. Land Ajacency of Europe



b. Region Adjacency Graph (RAG)



c. Equivalent RAG



d. Adjacency Matrix

Figure 4-19. Adjacency Graph and Matrix

RAGs are used frequently in image or picture processing to relate regions of homogeneous characteristics to neighboring regions. Many of the segmentation algorithms are based on RAGs. The RAG is simply stored for computer processing by an adjacency matrix. The adjacency matrix can be given values which are used for region growing or splitting. At the finest resolution, every single pixel can be considered a node of the RAG surrounded by its neighbors. Thresholding the pixel values (by color or intensity) allows grouping of like pixels into larger areas (region growing).

4.3.3.3 Line Adjacency Graph (LAG)

Most raster images are scanned one line at a time and recorded in some run coding scheme. Figure 4-20 illustrates a raster scan of a binary image of the printed character y. The scan pixel is equal in x and y so that the direction of scan has no influence on the recorded images. Each scan line is partitioned into segments where the image changes from black to white or vice versa. Nodes can be calculated which represent the center of each black segment. The nodes are connected by an arc to the previous line if:

1. The nodes are the same color

2. The segments overlap if projected onto a single line

The figure illustrates the Line Adjacency Graph created from a vertical and horizontal scan of the same line. Each graph is an unidentical skeletonized figure. Region boundaries can be calculated from LAGS by recording the ends of each segment (change points) and connecting the change points.

The LAG is effective for thin line pictures and can be used for other images by density slicing or thresholding before producing the LAG. In some implementations intra-line segments are also connected with the arcs' identified in a different manner. With intra-line arcs, the LAG becomes a special case of the RAG.

4.3.3.4 Hypergraph Based Data Structure

The Hypergraph (Bouille, 1978) is the most complex graph structure reviewed. It is a graph theory based general model which emphasizes the topologic properties of the data. The emphasis is on relations between four abstract data types: class, object, attribute, and relation.

The Hypergraph uses two types of graphs; a directed hierarchic graph (see Section 4.3.4 for hierarchical data structures) and a multigraph for the non-hierarchic portion. Links are added between classes and between

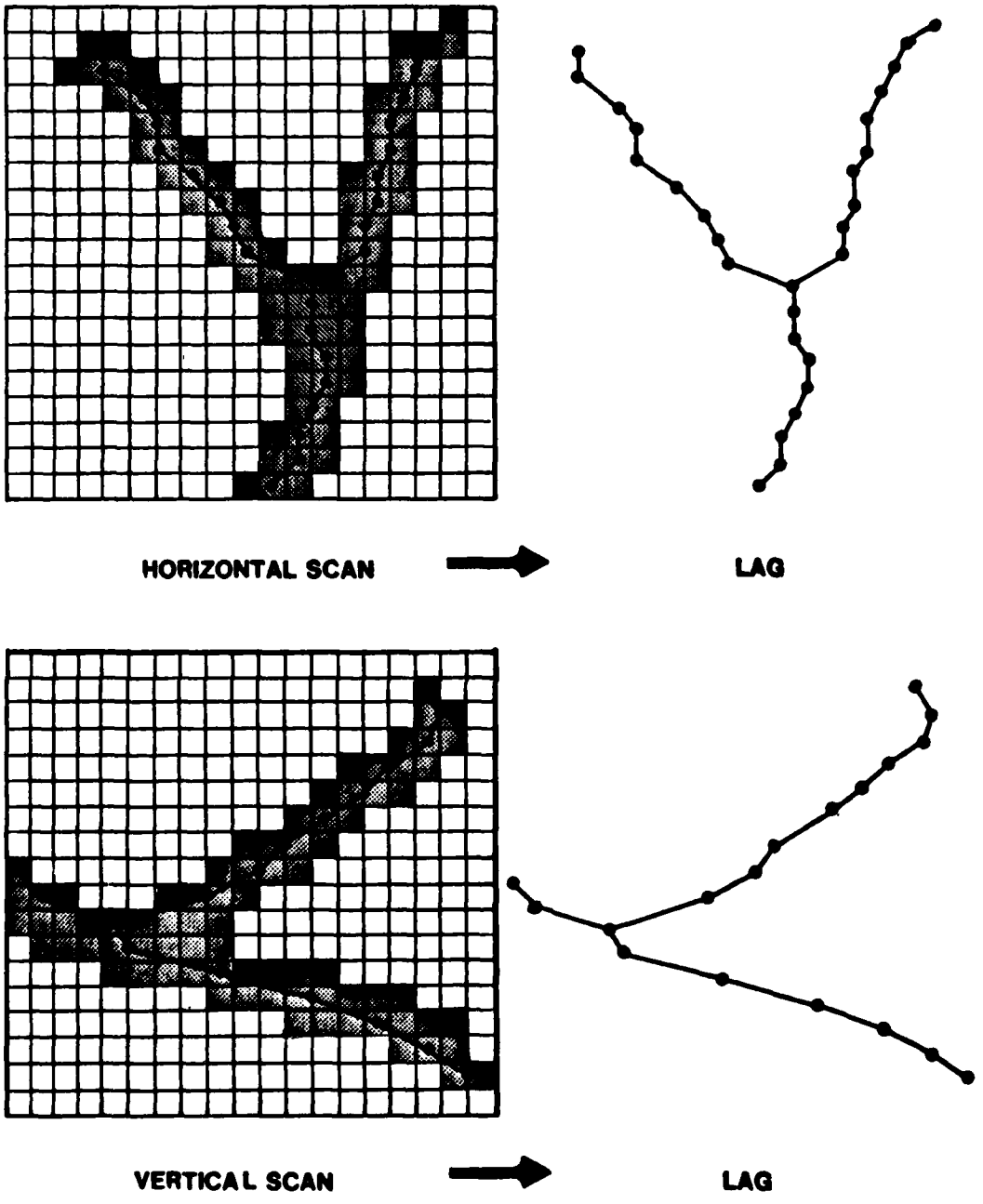


Figure 4-20. Line Adjacency Graph

objects to express additional relations. The basic structures of a hypergraph are shown in Figure 4-21. The basic structure is augmented with some extensions in the Hypergraph Based Data Structure (HBDS); hyperclasses, hyperlinks, multilinks and hypermultilink.

Hyperclasses are formed when all the classes belonging to a first group have links with all the classes of a second group and vice versa. The classes of group 1 and the classes of group 2 are then bundled into two hyperclasses.

Hyperlinks are the single link replacement for the multiple links when hyperclasses are formed.

Multilinks are a simplification when two classes are joined by several links. They are a multivalued link which replaces several links.

Hypermultilinks are a combination of the hyperlink and multilink concepts to simplify complex cases.

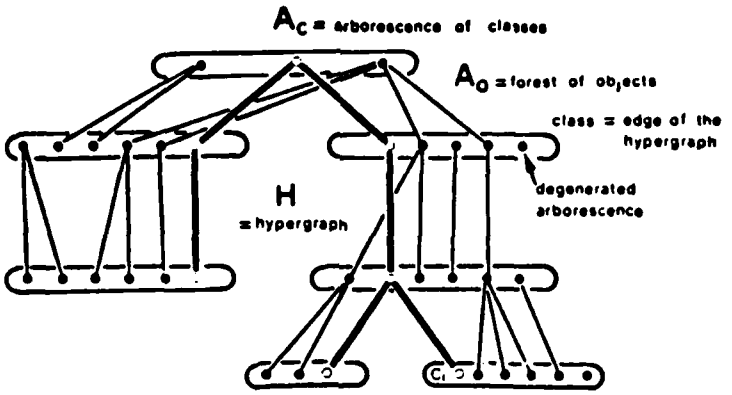
The HBDS has been applied to cartographic data to form a topologic based cartographic data bank of large scale. The implementation at the Universite Pierre et Marie Curie uses a simulation language (SIMULA 67) to define and manipulate the data. The HBDS shows promise for expressing topological properties.

4.3.3.5 Polygon Structure Graphs (PSG)

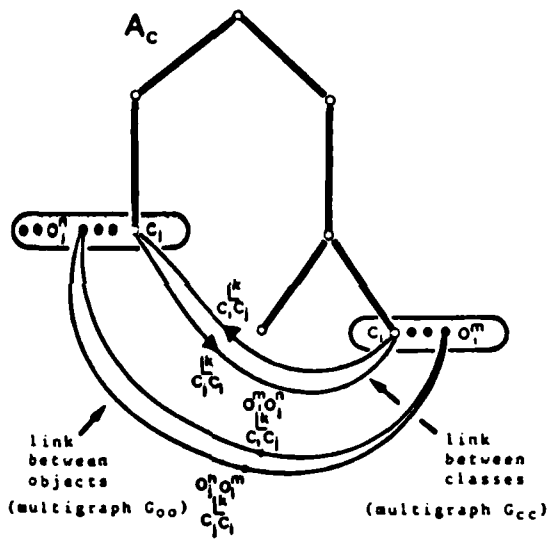
Polygon structure graphs (Tanimoto, 1981) are used for pattern recognition. The PSG encode structures in unit dimension or side to total ratios which allows comparison of detected shapes in an image to recorded shapes. The encoding of side lengths as unit dimensions permits comparisons between shapes of different scales. The graph itself is a combination of two graphs linked at a root node and via arcs. The first part of the graph has its first nodes the line segments followed by the included angle and a second line segment. The second half of the graph has the angle first, then the line segment and the next angle. Figure 4-22 shows a polygon and its graph structure. The sequence of aAbBc....AfA is numbered consecutively which facilitates tracing at the shape when the numbers are added to the graph as pointers.

The graph begins at the line segment a on the left and traces counter-clockwise. Lines are assigned a new letter when they differ in length from the previously recorded lines. Similarly angles are assigned a capital letter if not used before. In the figure the right angle ⌋ is assigned the letter A and repeated four more times.

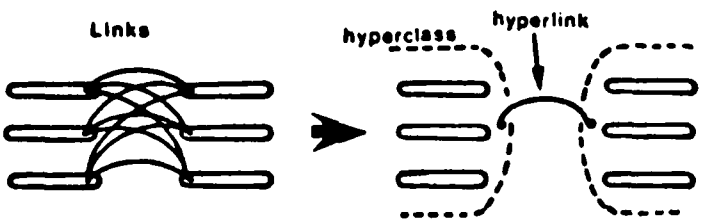
By using counter-clockwise descriptors, unit lengths and no orientation reference, the PSG is useful in identifying similar polygons.



THE HIERARCHIZED PART OF THE HBDS

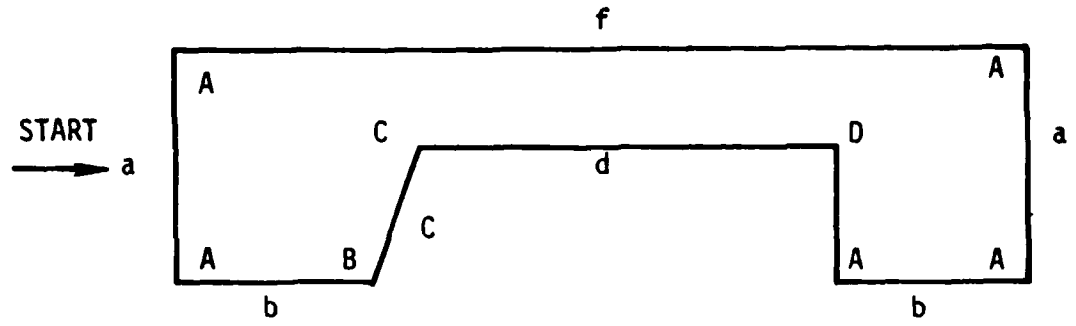


THE NON-HIERARCHIZED PART OF THE HBDS



HBDS EXTENSIONS: HYPERCLASS, HYPERLINK

Figure 4-21. HBDS



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
a	A	b	B	c	C	d	D	e	A	b	A	a	A	f	A

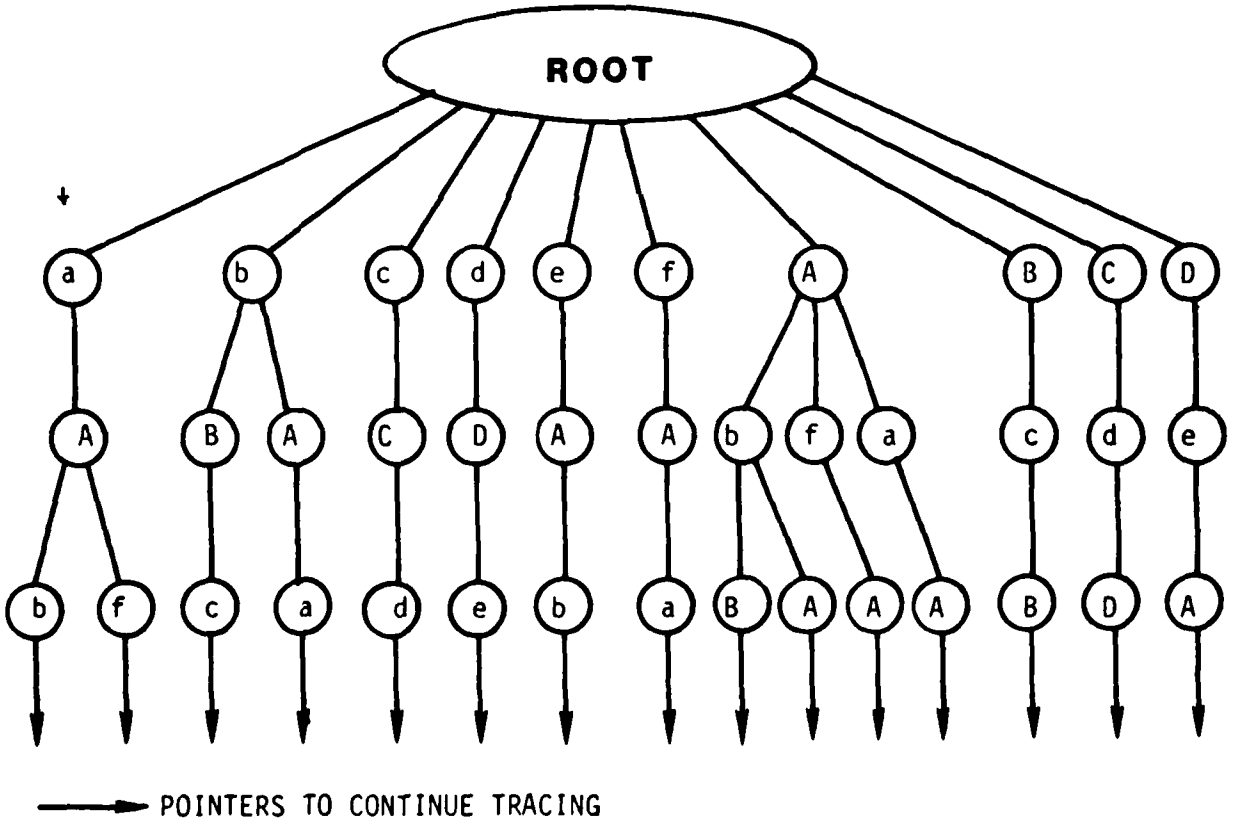


Figure 4-22. Polygon Structure Graph

4.3.4 HIERARCHIC DATA STRUCTURES

A common structure for representing pictures is the special graph called a tree; examples are given in Figure 4-23. A tree is a directed graphic without cycles, i.e., following a tree from the root (starting) node, there is no path or combination of arcs which allows a return to a node already visited. Many picture processing algorithms use trees to store information. The picture is divided into parts and the parts into subparts in a hierarchic structure. Algorithms for edge detection, region growing, segementation, relation labeling, texture analysis have used tree structures.

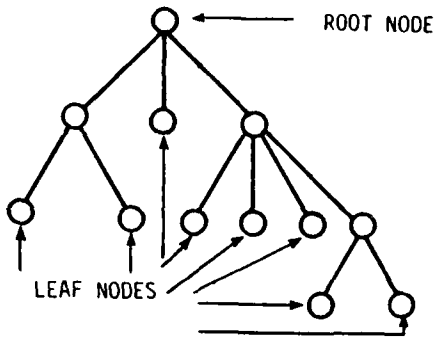
The concepts of inclusion and decomposition are the basis for segmentation. Figure 4-24 shows segmentation trees. Inclusion is a process which starts at some level and grows regions of the picture into larger regions by adding areas of similar characteristics. The concept of decomposition subdivides areas until areas are homogenous regions. The split and merge algorithms are implementations of decomposition and inclusion respectively.

The Quartic Picture Tree (QPT), as illustrated in Figure 4-25, is a form of hierarchic data structures. The leaf nodes of the tree are the pixels. The QPT is constructed from a root node with each subdivision of a node consisting of four children. Each node is some aggregate of its children, such as the average value. In the next section, we look at quad trees which are a form of QPT's. An extension of the quad tree is the rectangular region. Quad tree concept is then expanded to three dimensions in a structure called Oct trees.

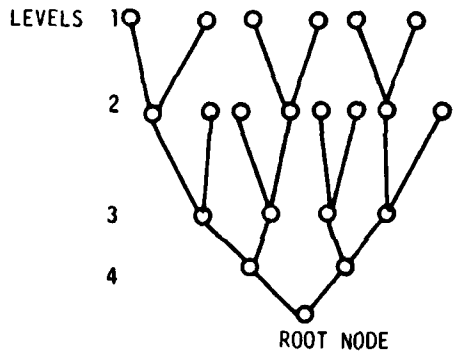
4.3.4.1 Quad Trees

In general, quad tree and Quartic Picture Trees are identical in structure except in some literature the QPT refers to the complete physical structure whereas the quad tree may be truncated where no new information is found at a lower level. The difference is shown in Figure 4-26. Note that in Figure 4-26a, the cut nodes are shown which divides the picture into regions. The quad tree structure requires less storage since all pixels need not be stored. Cut nodes are not usually stored in quad trees since they can be quickly calculated from the structure.

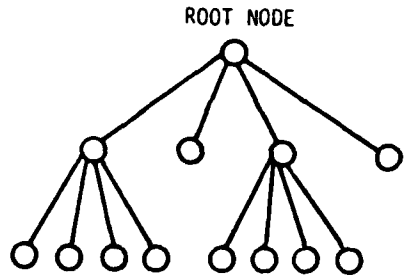
The quad tree and quartic tree have the advantage of supplying information at vaious levels of coarseness which improves the efficiency of some algorithms. Algorithms such as Horowitz and Pavlidis (Horowitz, 1976) Split and Merge Algorithm work from some central level and proceed in both directions to improve the result. The gross information at intermediate levels can be used like a first approximation with interactive procedures at other levels to refine the result.



A. GENERAL TREE

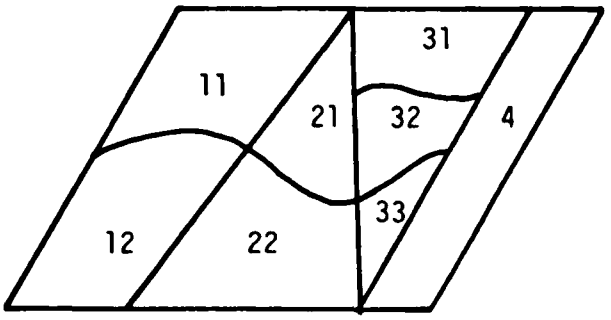


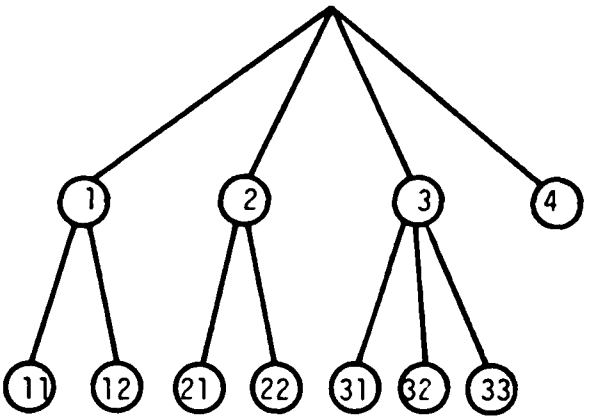
B. BINARY TREE



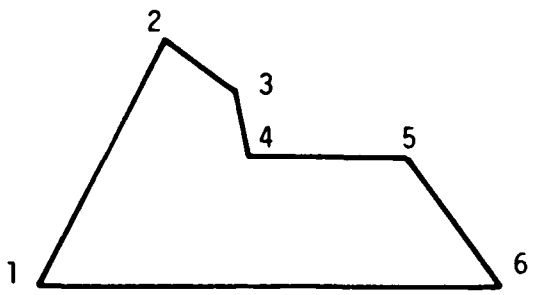
C. QUARTIC TREE

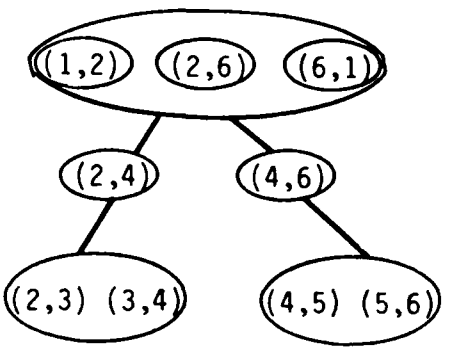
Figure 4-23. Tree Structures





SEGMENTATION **SEGMENTATION TREE**

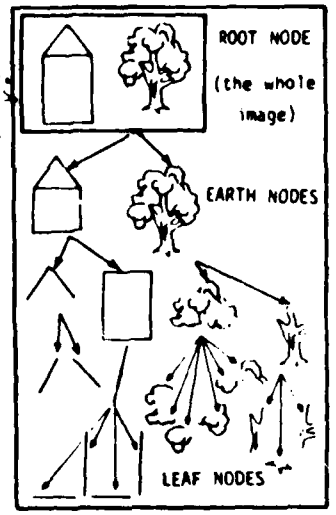




CURVE **CURVE SEGMENTATION TREE**

Figure 4-24. Segmentation Tree

IMAGES MAY BE CONSIDERED TO BE HIERACHICAL. A WHOLE IMAGE MAY BE BROKEN DOWN INTO COMPONENTS, THESE COMPONENTS INTO SMALLER COMPONENTS, AND SO ON DOWN TO A LEVEL OF PRIMITIVE ELEMENTS. THE DATA STRUCTURE USED TO REPRESENT HIERARCHIAL DATA IS A TREE. IMAGE COMPONENTS ARE REFERRED TO AS NODES, THE ROOT NODE BEING THE ENTIRE IMAGE, PARENT NODES BEING DIVISABLE COMPONENTS, AND LEAF NODES BEING INDIVISABLE COMPONENTS



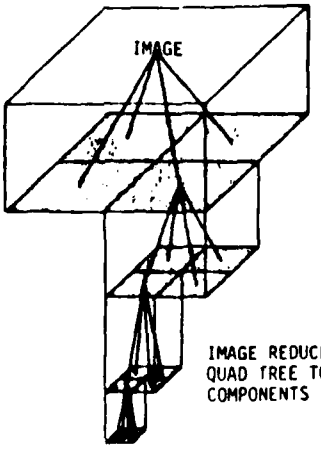
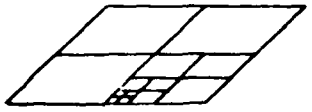
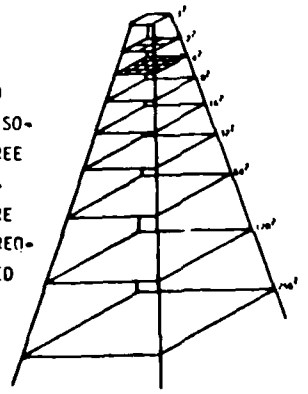


IMAGE REDUCED THROUGH QUAD TREE TO INDIVISIBLE COMPONENTS

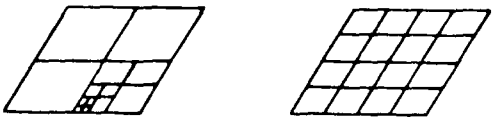


A SPECIAL VARIETY OF TREE IS THE QUAD TREE. A QUAD TREE IS A TREE WHOSE NODES ARE EITHER LEAF NODES OR PARENT NODES WITH FOUR CHILDREN NODES. WHEN REPRESENTING AN IMAGE, EACH LEAF NODE REPRESENTS AN INDIVISIBLE SEGMENT OF THE IMAGE AND BARES THAT SEGMENT'S COLOR. QUAD TREE ENCODING CAN BOTH PRODUCE MEMORY SAVINGS AND FACILITATE IMAGE ANALYSIS ALGORITHMS

THE PYRAMID OR PROCESSING CONE DATA STRUCTURE IS A HIERACHICAL REPRESENTATION OF AN IMAGE ORGANIZED INTO LAYERS, EACH SUCCESSIVE LAYER REPRESENTING A FINER RESOLUTION OF THE IMAGE. IT IS A RELATIVE OF THE QUAD TREE IN THAT EACH SUBDIVISION ON LEVEL N IS DIVIDED INTO 4 SMALLER SUBDIVISIONS ON LEVEL N+1. THIS DATA STRUCTURE IS VALUABLE WHEN VARIABLE RESOLUTION OF AN IMAGE IS REQUIRED, THAT IS, WHEN THE IMAGE IS DESIRED TO BE VIEWED AT VARIOUS LEVELS OF DETAIL. INFORMATION CAN FLOW UP, DOWN AND LATERALLY IN THIS STRUCTURE.

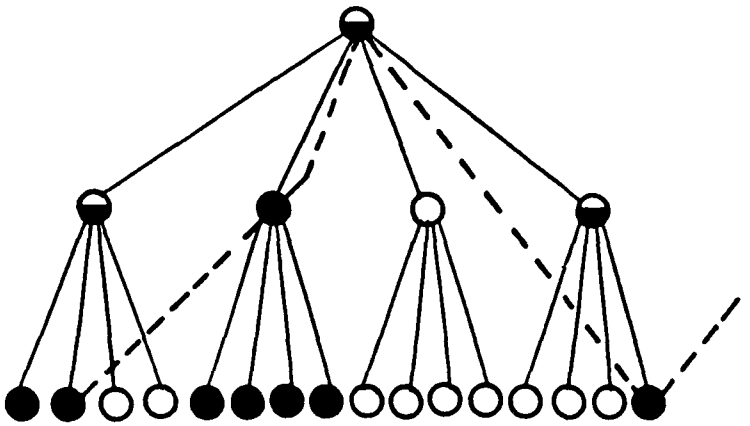


THE RASTER DATA STRUCTURE WHILE ESSENTIALLY NON-HIERARCHICAL POSSESSES SIMILARITIES TO BOTH QUAD TREES AND PROCESSING CONES. LIKE QUAD TREES, RASTER BASES SEGMENT IMAGES INTO FINER RECTANGULAR SUBDIVISIONS AND LIKE PROCESSING CONES, RASTER DATA CAN BE RESOLVED AT DIFFERENT LEVELS

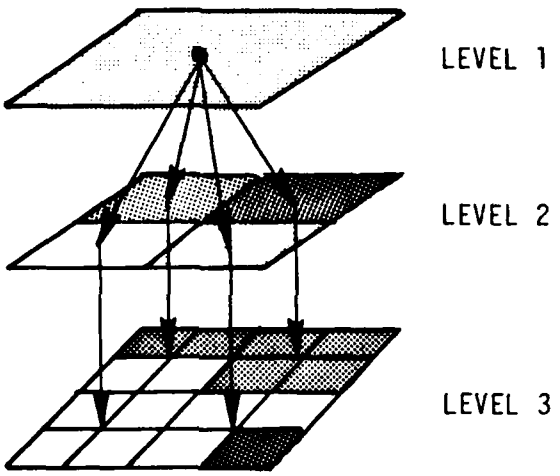


QUAD TREE RASTER BASE

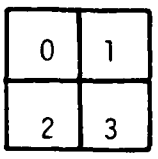
Figure 4-25. Quartic Picture Tree (Quad Tree)



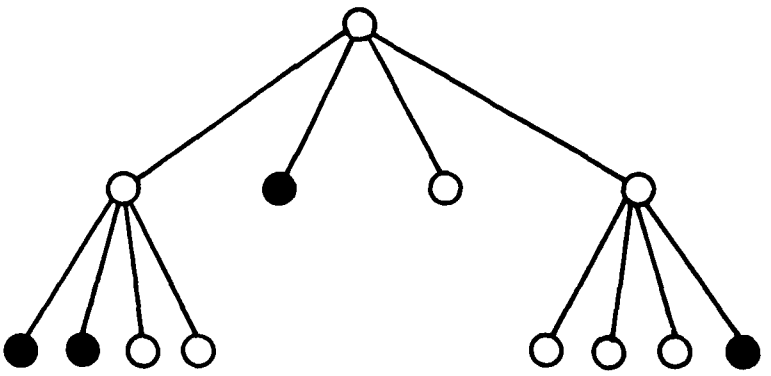
a. Quartic Picture Tree
(With cut nodes for black area)



n^2 picture n=4



QUADS



b. Quad Tree

Figure 4-26. Quartic and Quad Trees

4.3.4.2 Oct Trees

Oct trees are the three dimensional extension of the quad tree (Figure 4-27) used to store and display three dimensional objects. Sparse matrix storage techniques can be used to reduce the storage requirements of the 3-D representation. However, even with the sparse matrix techniques, the storage requirements are greater than those using polyhedra space fill techniques (not treated in this report). Polyhedra techniques use less storage since they describe large areas with a basic polyhedral shape defined in space. The Oct tree must use its unit building blocks to describe the shape by approximating the shape in a stepped sequence of cubes. The advantage of Oct trees is the ease of algorithmic applications for translation, rotation, region definition, and metrics. The Oct tree defines each unit block and its neighbors within the overall scheme.

4.3.4.3 Rectangular Regions

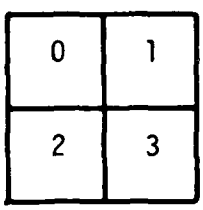
Another extension of the quad tree approach is to define regions by the largest rectangular block within the region and then add the remaining information. The structure is pyramidal with the gross information at a higher level and each subsequent level defining smaller rectangles until the last level contains the single pixels. Methods using rectangular regions have been developed primarily for transmission of binary images where large areas can be defined by a rectangle. Printed text lends itself to these techniques (see Figure 4-28). The border areas, space between lines, space between characters, and around the characters comprise much more than 50% of the page area. The advantage over quad tree descriptions is most obvious where long narrow rectangles can be defined.

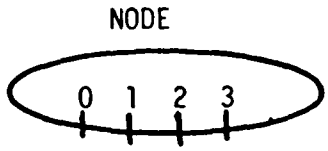
A second approach to Rectangular Regions is based on the nature of raster encoded data. Rather than defining the rectangular regions in terms of length and width, the regions are defined as a set of raster scans from a starting scan (Y_1) to a finishing line (Y_2) with segment lengths defined for each scan (X_1, X_2) or set of scans (Figure 4-29c). The methods are compared in Figure 4-29.

4.3.4.4 Hexagonal Structures

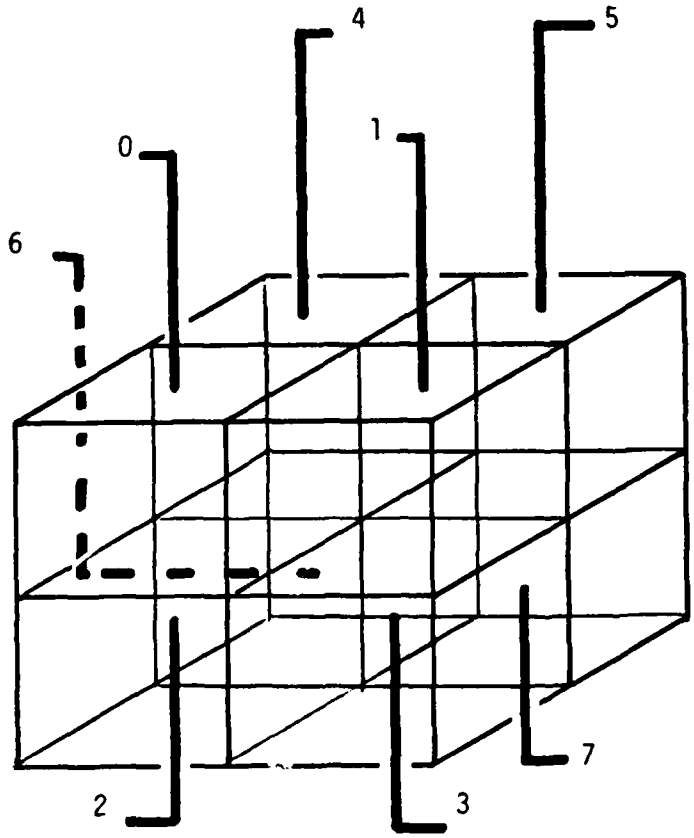
Although the prime description technique for surface coverage is the square grid, hexagonal grids can also be used as a uniform packed grid. The hexagonal grid can also be structured hierarchically and treated mathematically. The prime example is the GBT (Roach, 1980).

The General Balanced Thernary (GBT) data base structure technique is a method of representing a two-dimensional surface which assists computer representation of a complete picture as various levels of detail are required. The GBT method uses a geographically referenced, variable resolution





Quad Tree Node



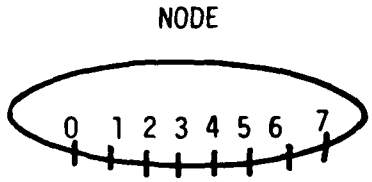


Figure 4-27. Oct Tree Node

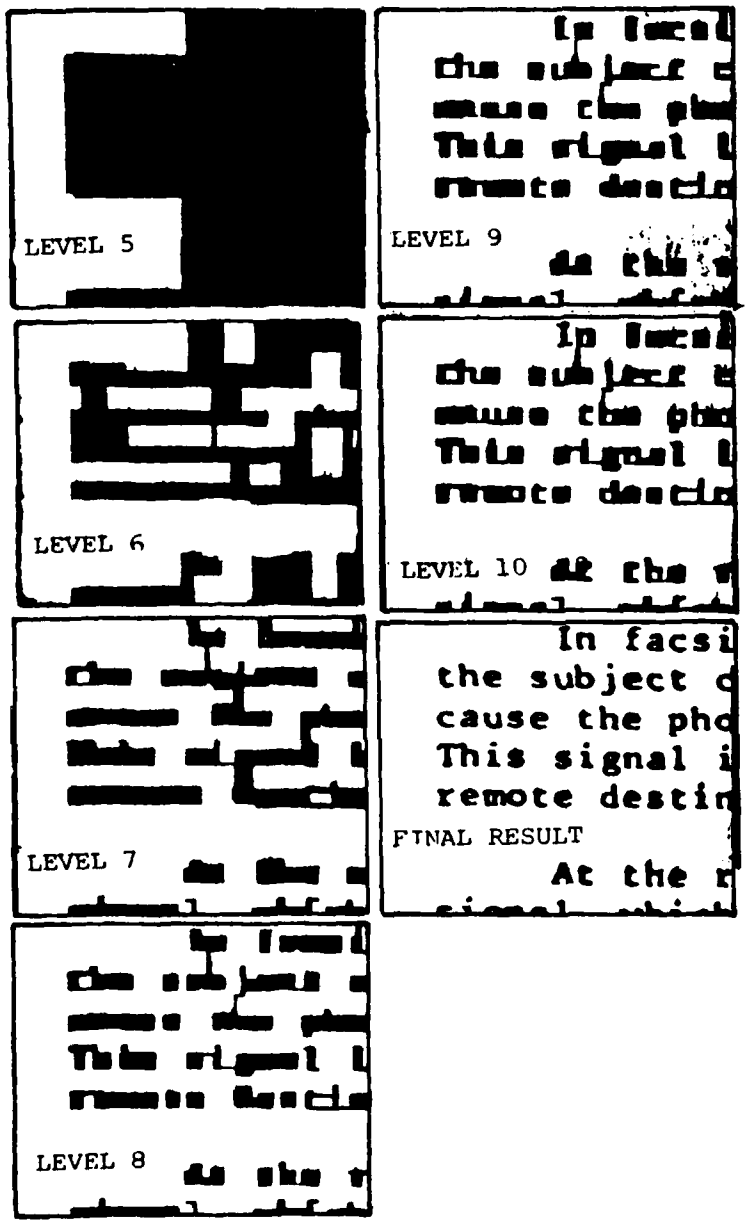
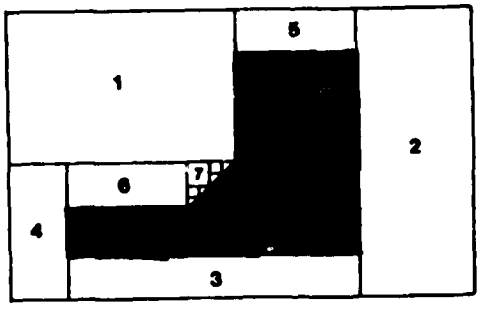
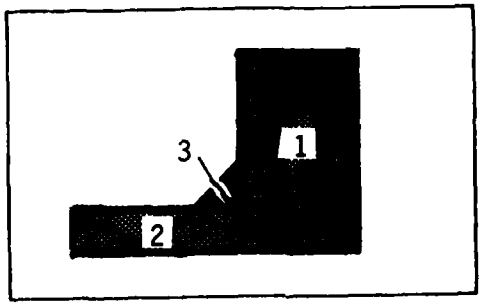


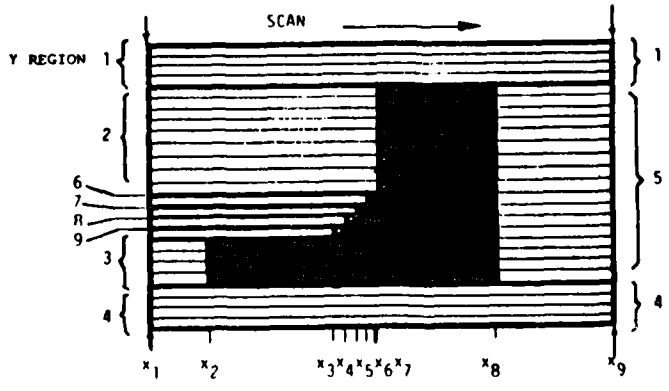
Figure 4-28. Rectangular Region Coding for Character Transmission Showing Regeneration at Subsequent Levels



A. DEFINITION BY WHITE AREAS



B. DEFINITION BY BLACK AREAS



C. DEFINITION BY RASTER SCAN AREAS OF WHITE

Figure 4-29. Rectangular Regions

data base and fast retrieval capabilities that provide an efficient means for examining data in aggregate form. The GBT structure permits layered accessing of finer and finer detail or alternatively the general content of the data can be examined at a high level without looking at the detail.

The GBT method uses a hexagonal cell structure in concert with a hierarchical addressing technique to permit aggregation.

The hexagonal coverage exhibits the property of uniform adjacency, which means that each cell shares exactly one-sixth of its boundary with each of six adjacent cells. The primary level-aggregate as shown in Figure 4-30 is a building block for the second-level aggregate and its neighbors.

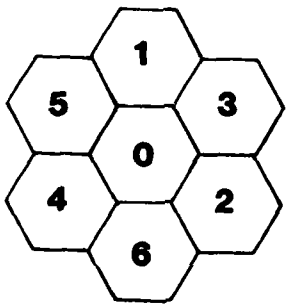


Figure 4-30. First Level Hexagonal Aggregate

Each hexagon has a unique GBT address, in which each digit corresponds to an aggregate level.

The GBT method includes an algebra for performing planar addition, subtraction, and multiplication upon the GBT addresses. A set of GBT addresses can be treated as a tree structure in which each address denotes a node on the tree. The tree structure provides an easy method for accessing data by general location.

In the GBT data structure data representations of Ø, 1, or 2 dimensions are described as points, vectors between points, and pointers to and from attribute files. Each point, line, and area is given a GBT reference address, which serves as a linkage pointer between the GBT georeferenced (locational) data base and the attribute files. The attribute files reference the features by using the GBT hexagonal addresses. Attribute files are associated in parallel to the GBT tree structure of the locational

data base. Hence, a region may be searched at a high level in the cellular hierarchy and the general description of that area can be determined without looking at the attribute files of subordinate levels. Thus the GBT method facilitates variable resolution display and when combined with algorithms provides a mechanism for automating visual image analysis processes and computer perception.

4.3.5 LINEAR LISTS

A list of data elements represents a set which can be defined in some manner. An order, or linear list can be defined as a set of linear ordered data elements (Giloi, 1978). Given this property there are many operations which can be performed on linear data sets. Such operations include the ability to determine the length of the list, reading in either direction and insertion, deletion, changing or addition of individual elements (Horowitz, Sahni, 1982).

Linear list data structures utilize the same fundamental principles of set theory. A data structure classified as a linear list involves some type of a single dimensional ordered set of elements. The most important aspect of linear list data structures is the order in which the elements area stored. They are used most often when the processing of data in a particular order is required. Within this category, the following data structures have been identified:

1. Run Coding
2. Chain Coding
3. Raster Scan Chain Code
4. Column Ending Notation
5. Linked Lists
6. Raster Encoded Polygon Structure
7. Sparse Matrices

The development of these data structures has revolved around the need for data compression. The capture of data in a raster form represents large volumes of data. To store and manipulate raster data, techniques have been developed which reduce the amount of information required to represent the data. Primary to these techniques are the data structures on which they operate.

4.3.5.1 Run Coding

Run coding is a simple line oriented data compression technique. When an image is stored in raster format, each scan line can be considered a segment of that image. The image can be detected by comparing adjacent pixels in the scanning sequence until a significant change is encountered. The change in detail indicates an "edge" has been detected (Pratt, 1978). The number of pixels between the first change detected and the second represents a "run" of like pixels. For example, the amplitude of the brightness of any pixel can be compared to a neighboring pixel. A significant difference would indicate they are not of the same feature and thus an edge has been detected.

There are two basic approaches to encode the data within run coding. The first method is to store the scan line (Y), followed by the starting X position of the run. The end position can be denoted by the number of pixels from the beginning of the scan line as shown in Figure 4-31. This approach is called run-end coding but has the disadvantage of requiring a large fixed number of bits to define each run position (Pratt, 1978).

A second and more efficient scheme is the run-length code. Again the scan line position and starting X position of the run is recorded. However, the end can be described in terms of the distance from the starting pixel. This approach requires fewer bits on the average compared to run-end coding.

Consequently, run-length coding offers the advantage of storing an image in a smaller amount of memory. Maximum storage efficiency is realized where images are made up of a few long runs. For example, realistic computer generated images can be stored in 10% of the space required with a traditional byte per pixel refresh buffer (Foley, Van Dam, 1982). This approach becomes a disadvantage where the run-length decreases thus increasing the storage requirements.

4.3.5.2 Chain Coding

As one of the most prevalent linear list structures, chain coding provides many advantages where the data represents a sequence of points. This structure has been used extensively for the representation of curves. The most simple approach to describing a curve is to record a series of X,Y pairs which give the exact location of each pixel. However, this is extremely inefficient as it requires large amounts of storage. The basic chain code offers improved efficiency by recognizing that a single raster pixel has only eight possible neighbors. This can be applied to curve representation by encoding each point relative to a previous point.

For example, a grid can be superimposed over a curve. The curve can be described by a list of coordinates where the grid is intersected by the

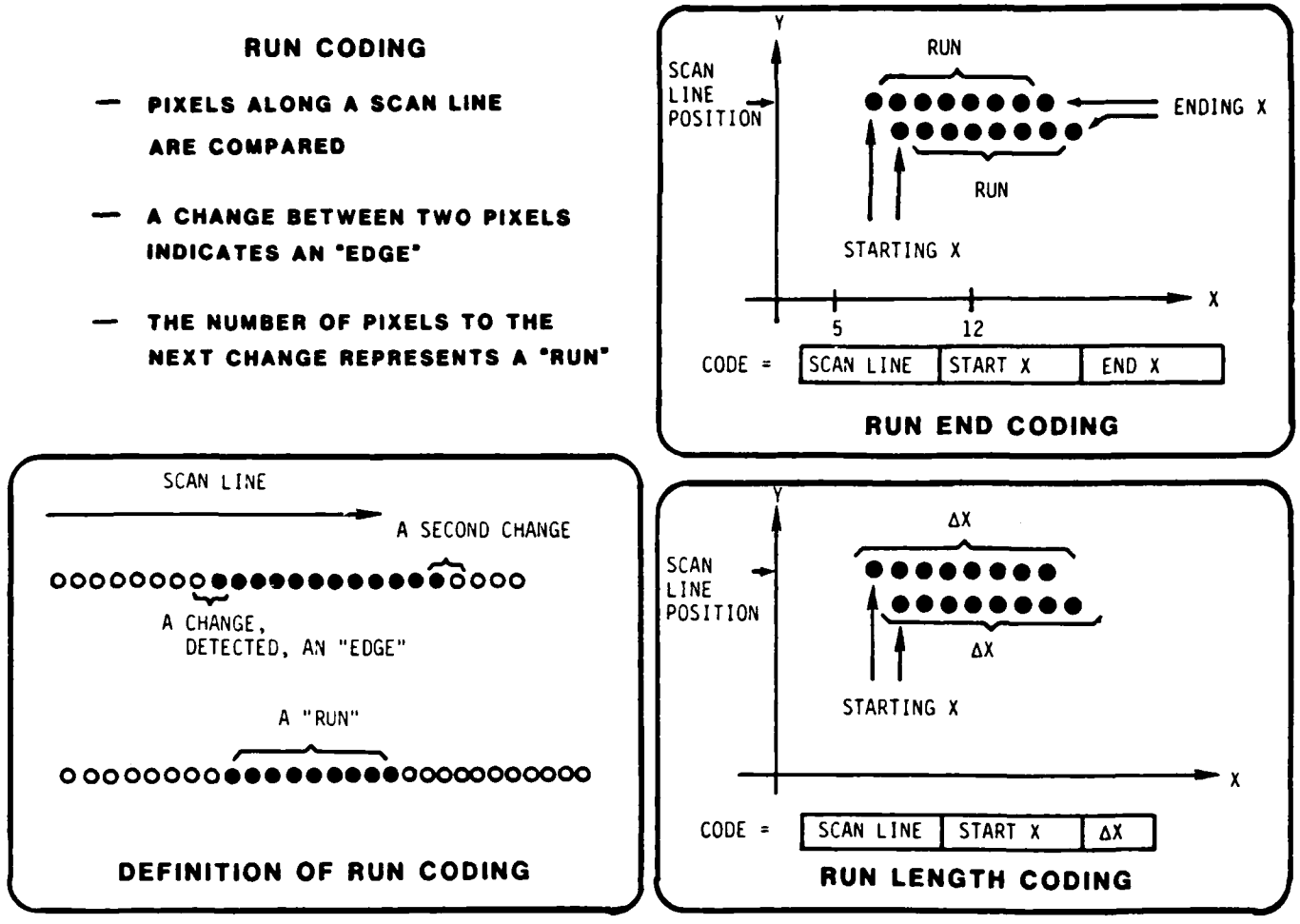


Figure 4-31. Run Coding

curve. Each coordinate can then be encoded relative to a previous point in terms of direction (Shapiro, 1979). Hence we can represent an entire curve by an initial X,Y position followed by a sequence of directions to the following points (see Figure 4-32). Since only three bits are needed to store the direction, this scheme provides substantial savings in required storage.

Several variations of the basic chain code have been offered to improve efficiency. One of these is a differential chain code where points are represented by a difference between two successive absolute points (Pavlidis, 1982). The number of directions is the same as the basic chain code but are given the values; Ø, $\pm$1, $\pm$2, $\pm$3, 4. It can be seen that for smooth curves the values Ø, $\pm$1 occur more frequently. This fact makes it possible to utilize a variable length encoding scheme with the differential chain code (see Figure 4-32). Pavlidis has found that such an encoding usually requires no more than two bits per point on the average. Primary applications of this encoding scheme are alphanumeric characters and outlines of objects where 1.5 to 1.9 bits on average are required. Curve representation by this encoding scheme may not be desirable depending upon the application.

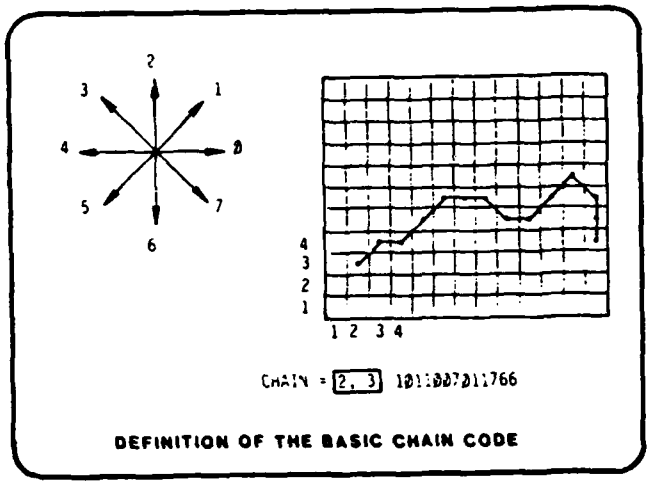
Two additional variations on the differential chain code has been described by Baudelaire and Stone. The first one is based on the concept of quadrants and uses two bits to represent the differential increment. This scheme divides the eight possible curve directions into four quadrants represented by a Ø, 1, 2, or 3. Within each quadrant there are three possible directions or increments which are assigned the values 1 to 3 (see Figure 4-33). The encoding of a curve would start with a quadrant number (Ø to 3) followed by the increment codes (1 to 3) and terminated by a Ø.

The second scheme offered by Baudelaire and Stone divides the set of eight possible directions into eight quadrants. Within each quadrant there are only two possible directions which can be represented by one bit. Two bit streams are used, one indicates the octant followed by the number of one bit increments. The second holds the actual one bit increments. This method offers the advantage of understanding the behavior of a curve by examining the octant codes alone. Edge definition for a scan converter is an example of where this feature can be applied.

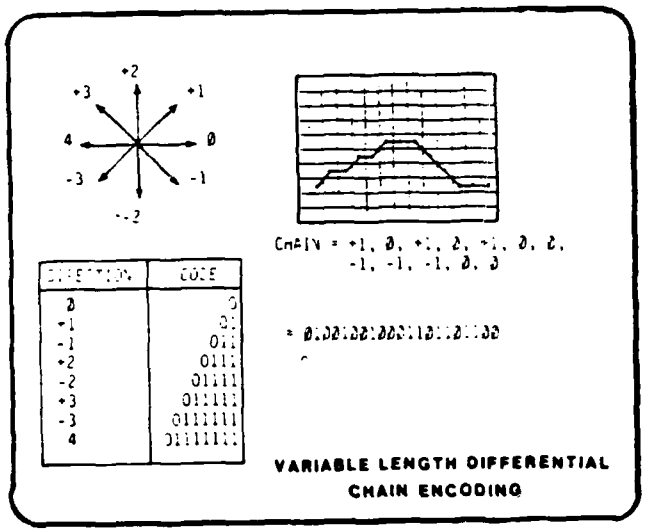
Further research into chain coding by Freeman has produced higher order encoding schemes. Based upon the concept of the eight point code, sixteen, twenty-four, and higher encoding schemes are possible (see Figure 4-32). These higher order chain codes appear to provide potential advantages to cartographic data. The desirable characteristics provide improved efficiency in storage, smoothness, and reduced processing time. Freeman evaluated these encoding schemes with respect to:

CHAIN CODING

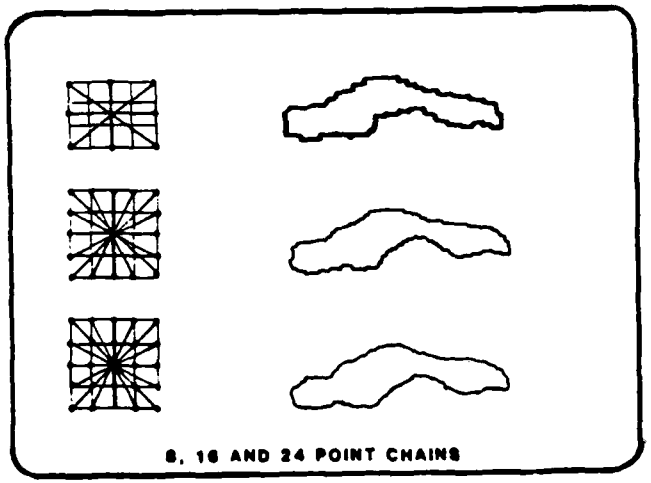
- A POINT CAN BE ENCODED RELATIVE TO A PREVIOUS POINT
- DATA REPRESENTED BY A SEQUENCE OF DIRECTIONS



DEFINITION OF THE BASIC CHAIN CODE



VARIABLE LENGTH DIFFERENTIAL CHAIN ENCODING



8, 16 AND 24 POINT CHAINS

Figure 4-32. Chain Coding

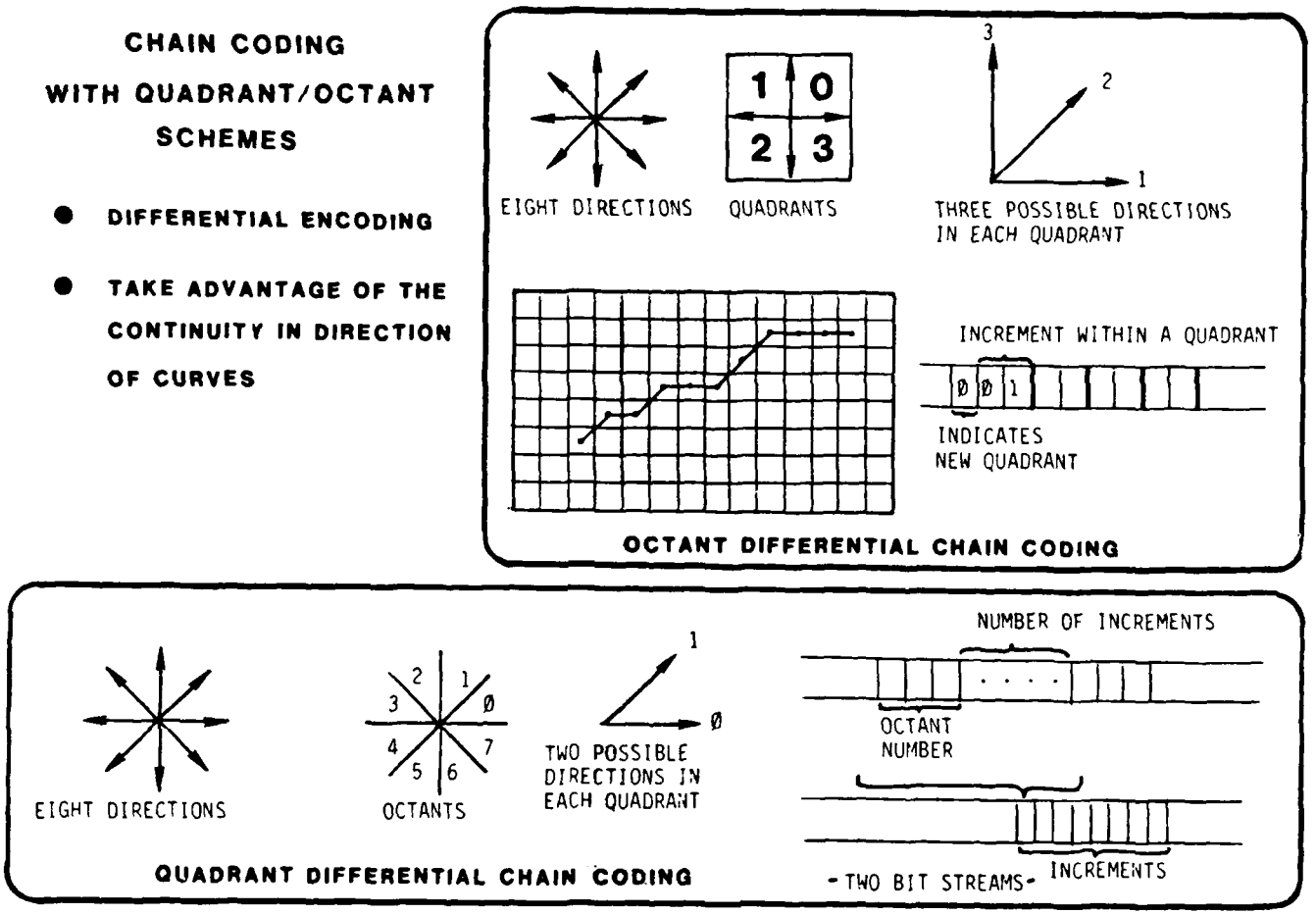


Figure 4-33. Chain Coding With Quadrant/Octant Schemes

1. Precision

2. Compactness

3. Simplicity of Encoding

4. Facility for Processing

5. Smoothness

His analysis found that higher order chain codes permit a smoother representation in a more compact form with less processing time. However, this does not occur without cost. The advantages must be weighed against increased complexity of encoding the data.

4.3.5.3 Raster Scan Chain Code

Chain and run coding provide a reduction in the amount of explicit information needed to represent an image. However, both of these techniques require the image to be raster digitized and then encoded from intermediate storage. A promising data structure which overcomes this disadvantage has been described by Roger L.T. Cederberg. His research at the National Defense Research Institute in Sweden has developed a data structure which is adapted to raster scanning devices. Binary images can be immediately segmented and encoded without intermediate storage and with higher code efficiency. Its unique characteristics further allow for detection of objects and holes and hierarchical storage all within the raster scanning process.

With chain coding, there are eight possible directions to the next pixel in the chain. To retrace the image the entire image needs to be accessible. An intial X,Y coordinate is read followed by a sequence of directions. If the image is viewed in the same manner as raster scanning device, then only four directions are possible (see Figure 4-34). Thus the potential for further data compression over the basic chain code can be seen.

Unlike chain coding, however, the raster scan chain code requires several coordinates to be recorded for each image. These coordinates correspond to the first pixels that a raster scan sequence encounters on some part of the image. These first bits are called max points and will be connected to two chains. Collectively the max points and associated chains will describe an image. Cederberg also defines min points as being the terminal point of two chains (Figure 4-34). Thus a single chain

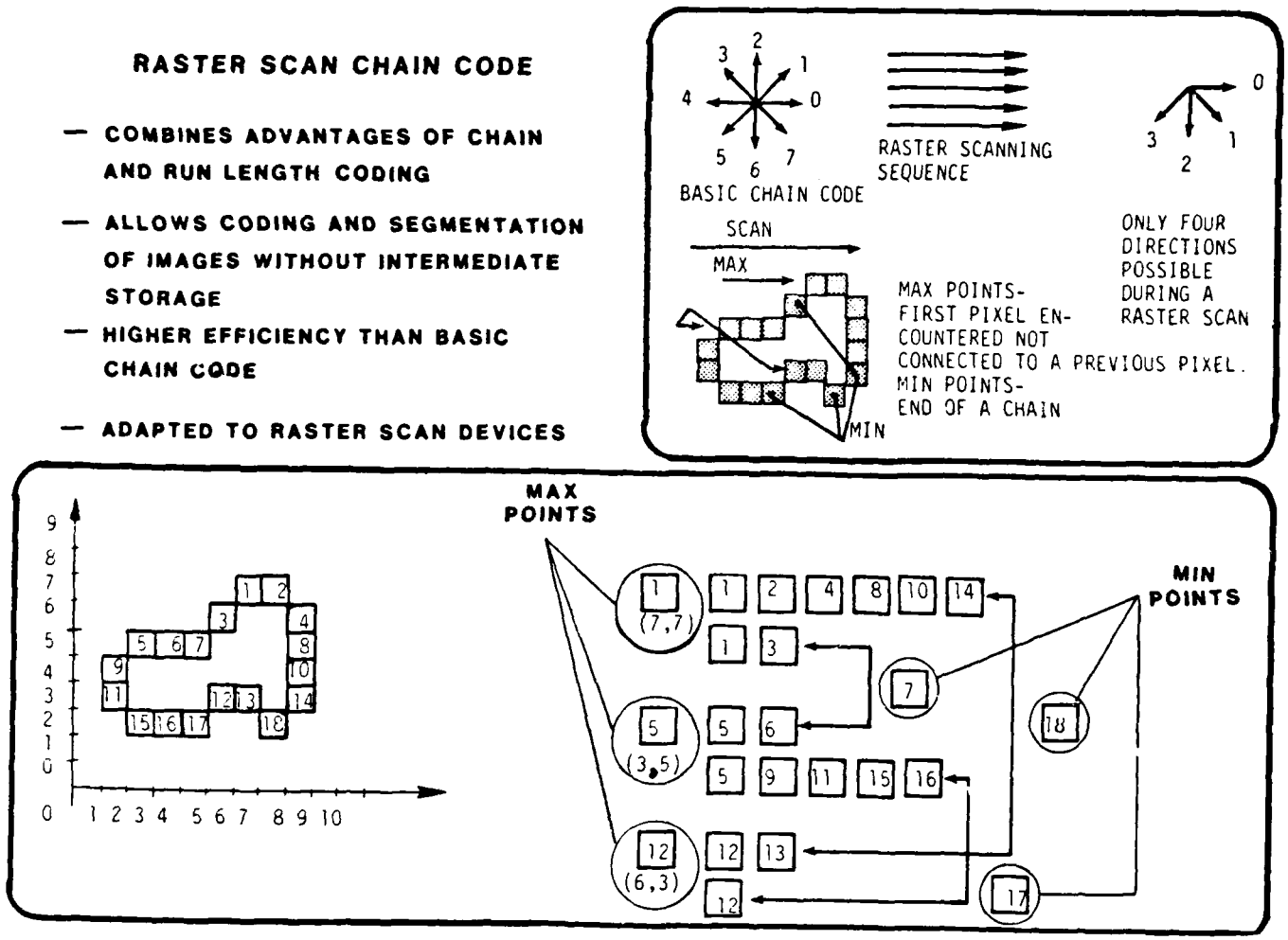
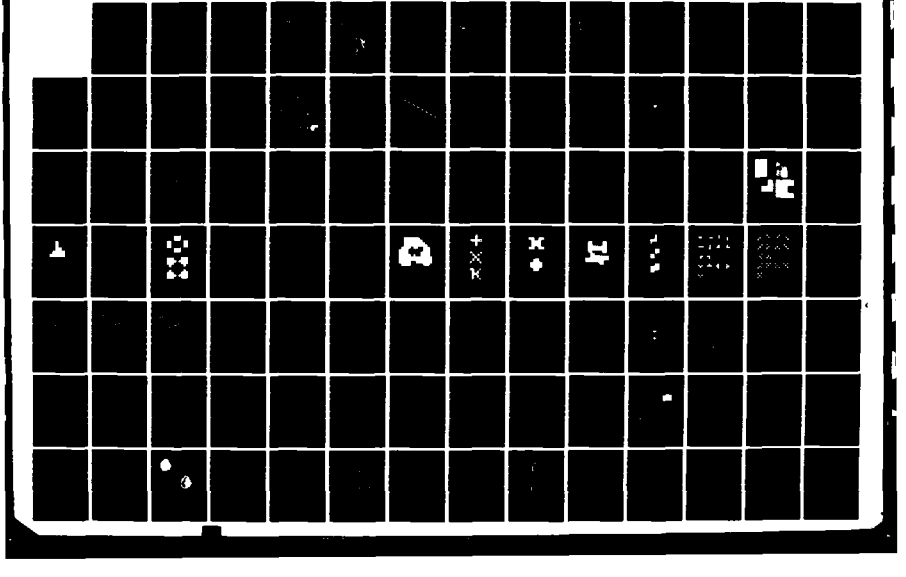
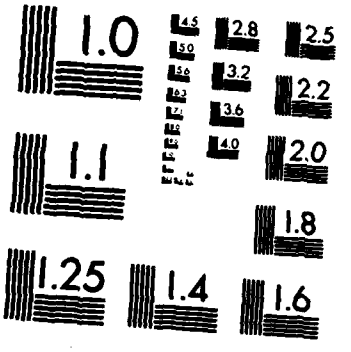


Figure 4-34. Raster Scan Chain Code

AD-A130 195 INTERACTIVE RASTER DATA STRUCTURE STUDY(U) SYNECTICS 2/3
CORP ROME NY D R GALLAURESI ET AL. JAN 83
RADC-TR-83-12 F30602-81-C-0279

UNCLASSIFIED F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

represents a segment of the image and must be linked to other chains to recover the entire image. However this does not require additional processing due to the pre-recorded min points which define the junction of two chains. Additionally, access to the entire image is not required. This permits processing of individual segments without the need to hold the subsequent chains.

Operation of the raster scan chain code utilizes the interconnections to combine the separate chains into a single string of links (Cederberg, 1978). The procedure begins with an initial coordinate (max point) for which one of the associated chains is read. Completion of one chain leads to a min point and a pointer to the end of another chain. This next chain is read backwards until its' coordinate (max point) is read, followed by a pointer to the second chain. This procedure is followed until the first coordinate is retrieved. Two additional lists are created simultaneously for the purpose of redrawing the image in a raster mode. This storage algorithm permits entities to be accessed as raster data and as regions, boundaries, and lines.

In brief, this data structure has potential advantages for the raster cartographic environment. The capability to digitize an image with a raster scanner and immediately segment and encode it without intermediate storage has great potential. Further possibilities that this technique could provide are shown in Figure 4-35.

4.3.5.4 Column Ending Notation

Another data compression technique is the column ending notation being researched by Stephen Miller at the U.S. Geologic Survey. This technique is similar to a run-end code because it is line oriented and segments the data. The difference is that run-coding can describe a single image on a raster row, whereas column ending notation describes the entire raster row.

Each raster row is identified by an initial header record with a -1 in the first field. The remaining fields could store other information such as row number. The subsequent records are a fixed format and represent compact data strings. The data strings are composed of a minimum of one attribute and a corresponding ending column position. For example, the first data string following the header record would have the initial attribute of that row and in which column on the row it ended. The next data string would begin with a different attribute followed by its corresponding ending position. This is depicted in Figure 4-36.

Unique to this method, no starting column position is necessary. A starting location can easily be derived by adding one to the previous ending position. Thus this represents a way to minimize the explicit

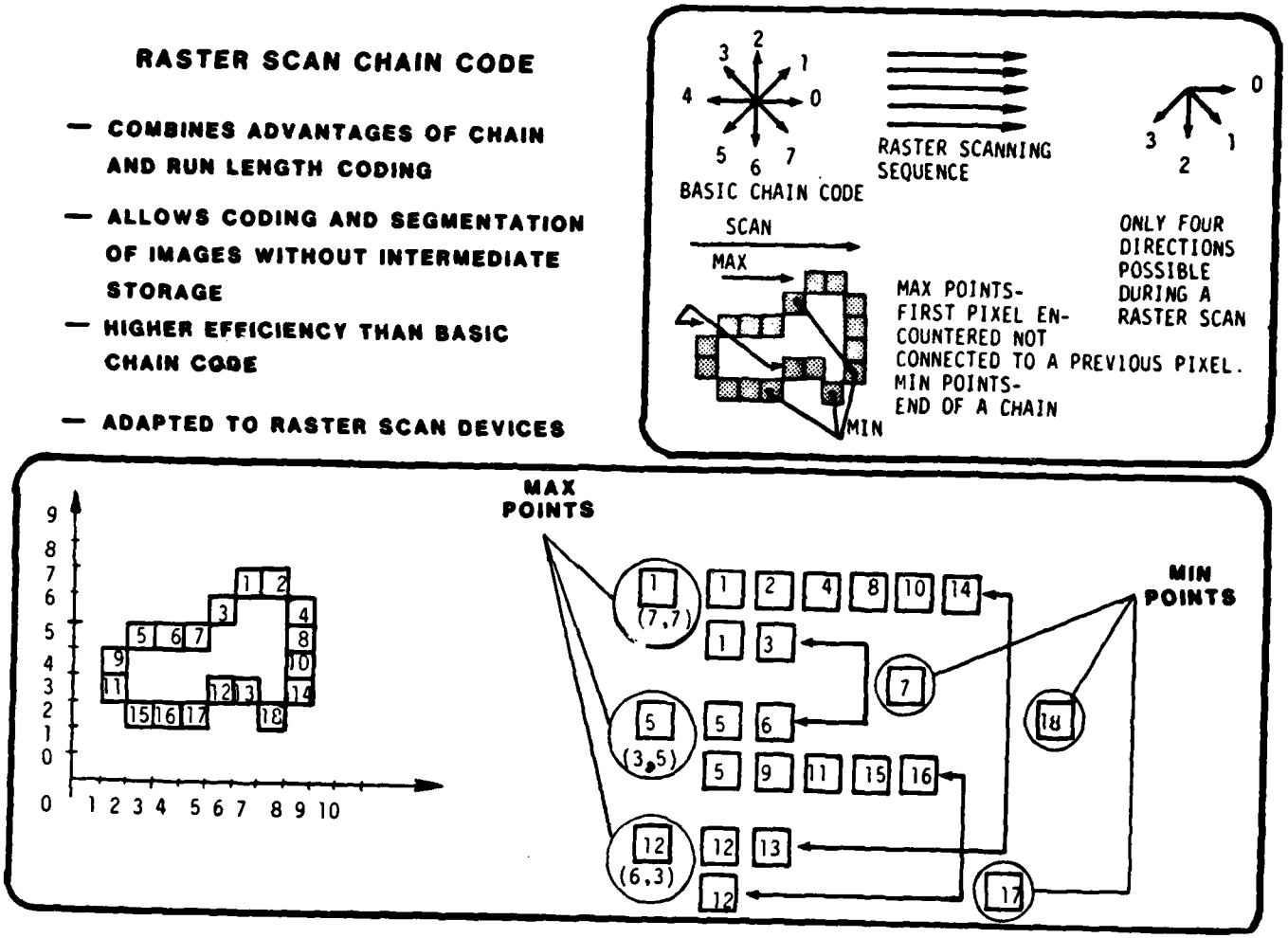
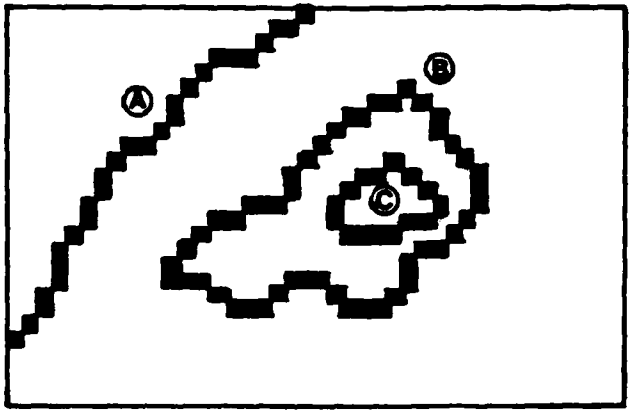


Figure 4-34. Raster Scan Chain Code

RASTER SCAN CHAIN CODE

POSSIBLE INFORMATION THAT COULD BE DERIVED AS A RESULT OF THIS TECNIQUE

- DETECTION OF IMAGES, OBJECTS OR HOLES
- THE NUMBER OF IMAGES OR OBJECTS
- OPEN VERSUS CLOSED IMAGES
- THE BOUNDS OF IMAGES OR REGIONS
- THE RELATIONSHIP OF IMAGES TO OTHER IMAGES



- THERE ARE A TOTAL OF THREE IMAGES
- B AND C ARE CLOSED, A IS OPEN
- C LIES WITHIN B
- B AND C ARE BELOW A

EXAMPLE

Figure 4-35. Raster Scan Chain Code

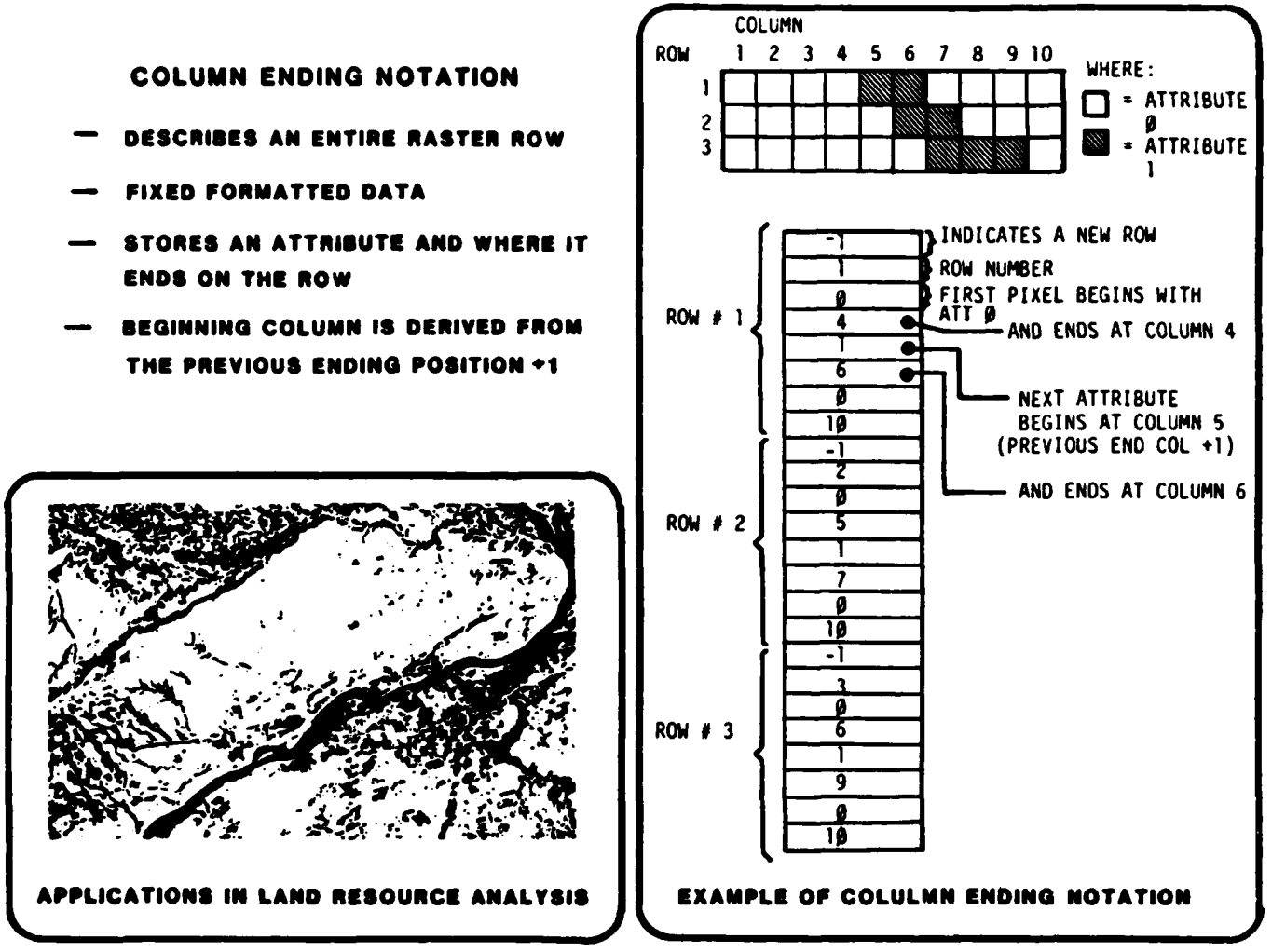


Figure 4-36. Column Ending Notation

recording of data. It has been applied to land resource analysis where a great deal of redundancy in the data exists. However, the properties unique to this data structure can also pose a potential threat. When data is sparse it may not be efficient since it is necessary to describe the entire raster row.

4.3.5.5 Linked Lists

A linked representation of a linear list is called a linked list (Giloi, 1978). Linear linked lists are usually composed of fixed records or fields which store a data element and, at minimum, a pointer to the next link. A single pointer list with a null pointer at the end is defined as a single linked list. This type of list is unidirectional and must be processed from start to finish. Other variations of linked lists are shown in Figure 4-37.

Circular linked lists take the single linked list one step farther. Instead of a null pointer at the end, the last record points to the beginning of the list. This offers the advantage of accessing the data starting at any position. Even more efficiency can be realized with a double linked list. This structure utilizes a second pointer along with the advantage of a circular linked list. The second pointer provides a link back to the previous record. This allows data elements to be accessed in either direction beginning at any record. Double linked lists also increase the efficiency of maintaining the list.

In relation to raster data, linked lists do not provide efficient data compression. However, the concept might be used in conjunction with other data structures. For example, a linked list structure could be used to store attribute codes and pointers to where the attributes begin. The data could then be stored as a run length code. Linked lists can also be used in the representation of sparse matrices.

4.3.5.6 Raster Encoded Polygon Structure

This data structure was developed for the Domestic Information Display System (DIDS). The primary purpose of this system is the processing of geopolitical data organized and displayed in map format. To efficiently process and display full raster images a data structure was developed which draws upon the raster orientation of color graphic displays and the storage efficiency of polygon data structures (Dalton, Winkert, Quann).

With run-length encoding, an image can be segmented by scan line. Each scan line for a geopolitical map display can be described by a series of nodes. Each node would contain a starting pixel coordinate followed by an area identifier. However this requires that all lines be processed in order to display the map.

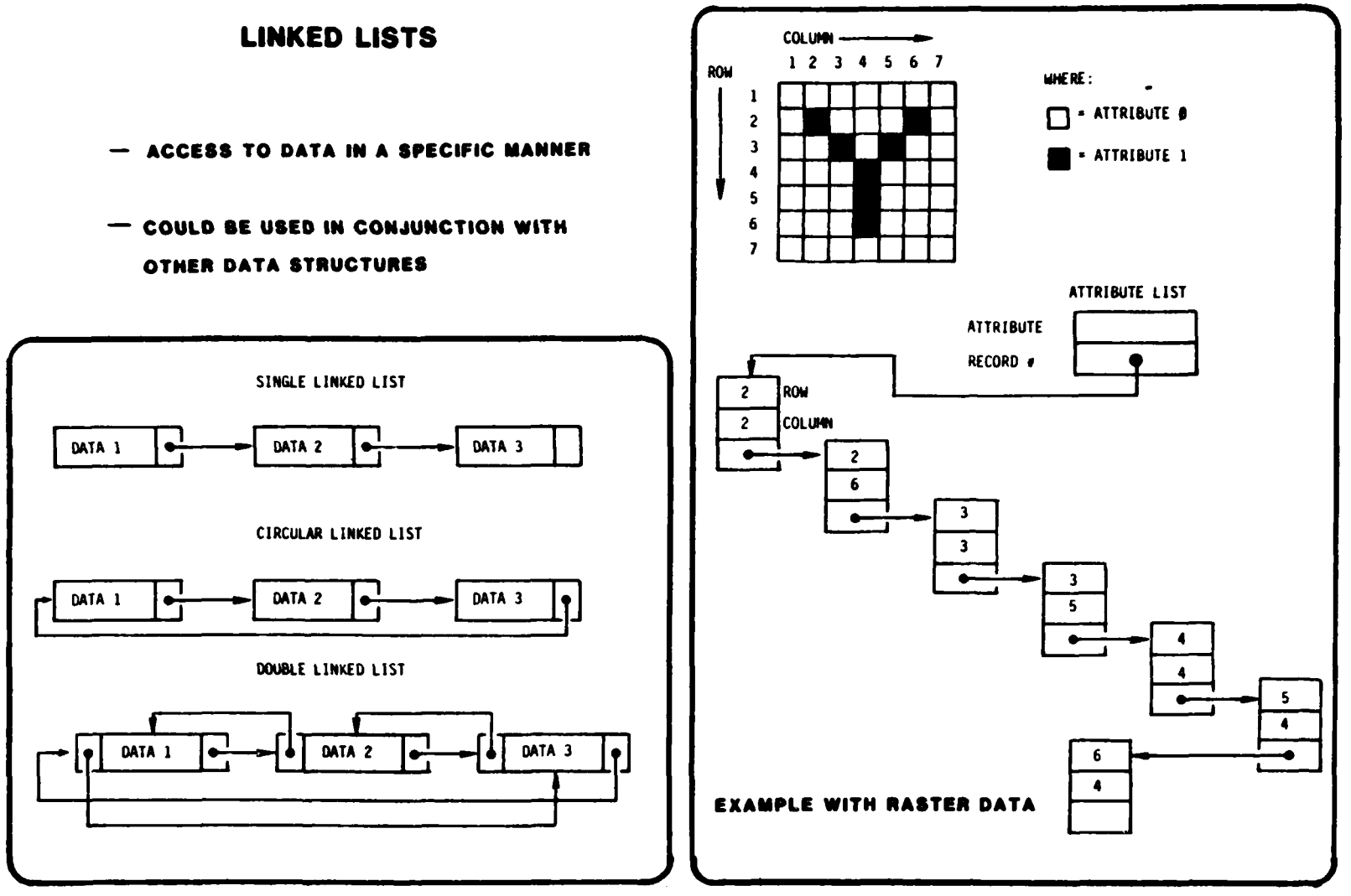


Figure 4-37. Linked Lists

The raster encoded polygon structure is a modification to the run-length encoding scheme. In addition to a starting pixel coordinate and area identifier, the number of scan lines upward to state and country boundaries are recorded. A flag indicating a state change from the previous node is also stored. This node definition is shown in Figure 4-38.

This structure does not provide any further data compression over run-length encoding. However it does provide a decrease in the amount of processing time required to display images. At any one time all the information needed to process and display an image at any scale factor is available. Thus, a 512 x 512 screen can be manipulated without referencing elements outside of the 512 x 512 area.

4.3.5.7 Sparse Matrices

A matrix generally consists of rows and columns that can be easily stored as a two dimensional array. This permits quick access to any data element of the matrix. If examination of the matrix finds many zero elements, then the matrix can be described as sparse. No clear definition exists as to when a matrix is sparse and when it is not. One way of describing a sparse matrix is in terms of its density. Matrix density is defined as the number of non-zero elements in the matrix (Pooch and Neider, 1973). Still, the variations in accepted density ranges leaves the definition unclear. However it can clearly be seen that a matrix with fifteen non-zero elements out of one hundred is definitely sparse.

In reality, matrices are usually larger than one hundred elements. Storage of an entire sparse matrix would be extremely inefficient. Hence there exists a need to compress the data in the matrix. One alternative is to store the non-zero elements along with the row and column index for each element (Horwitz, Ellis 1976). The matrix would then be represented as a list of three-tuples in the form of: (row, column value). Organization of this list by increasing columns within increasing rows can aid in processing. (Figure 4-39). Other techniques which have been developed for sparse matrices include:

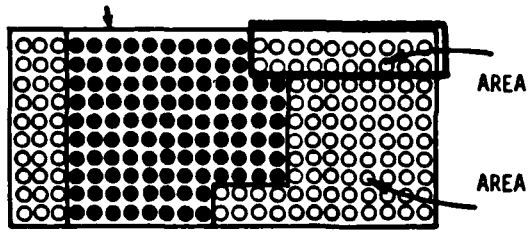
1. Bit map
2. Address map
3. Threaded or Linked lists
4. Band indexing

The following discussion provides a brief overview of these additional indexing techniques as described by Pooch and Neider.

RASTER ENCODED POLYGON

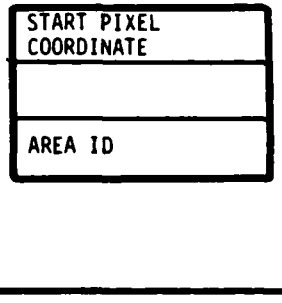
— **MODIFICATION OF RUN LENGTH CODING**

— **STORAGE MEANS FOR RAPID DISPLAY OF GEOPOLITICAL MAPS**

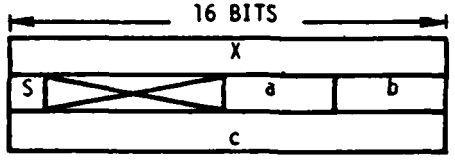


- AN IMAGE CAN BE SEGMENTED BY SCAN LINE
- EACH SEGMENT OF THE IMAGE WILL HAVE THE SAME AREA IDENTIFICATION
- A NODE WILL DEFINE A STARTING X COORDINATE AND AREA IDENTIFICATION

RUN-LENGTH NODE DEFINITION:



RASTER ENCODED POLYGON NODE DEFINITION:



X: STARTING PIXEL COORDINATE

S: INDICATES STATE CHANGE FROM PREVIOUS NODE

a: NUMBER OF LINES UPWARD TO STATE BOUNDARY

b: NUMBER OF LINES UPWARD TO COUNTRY BOUNDARY

c: COUNTRY I.D.

Figure 4-38. Comparison of Run-Length and Raster Encoded Structures

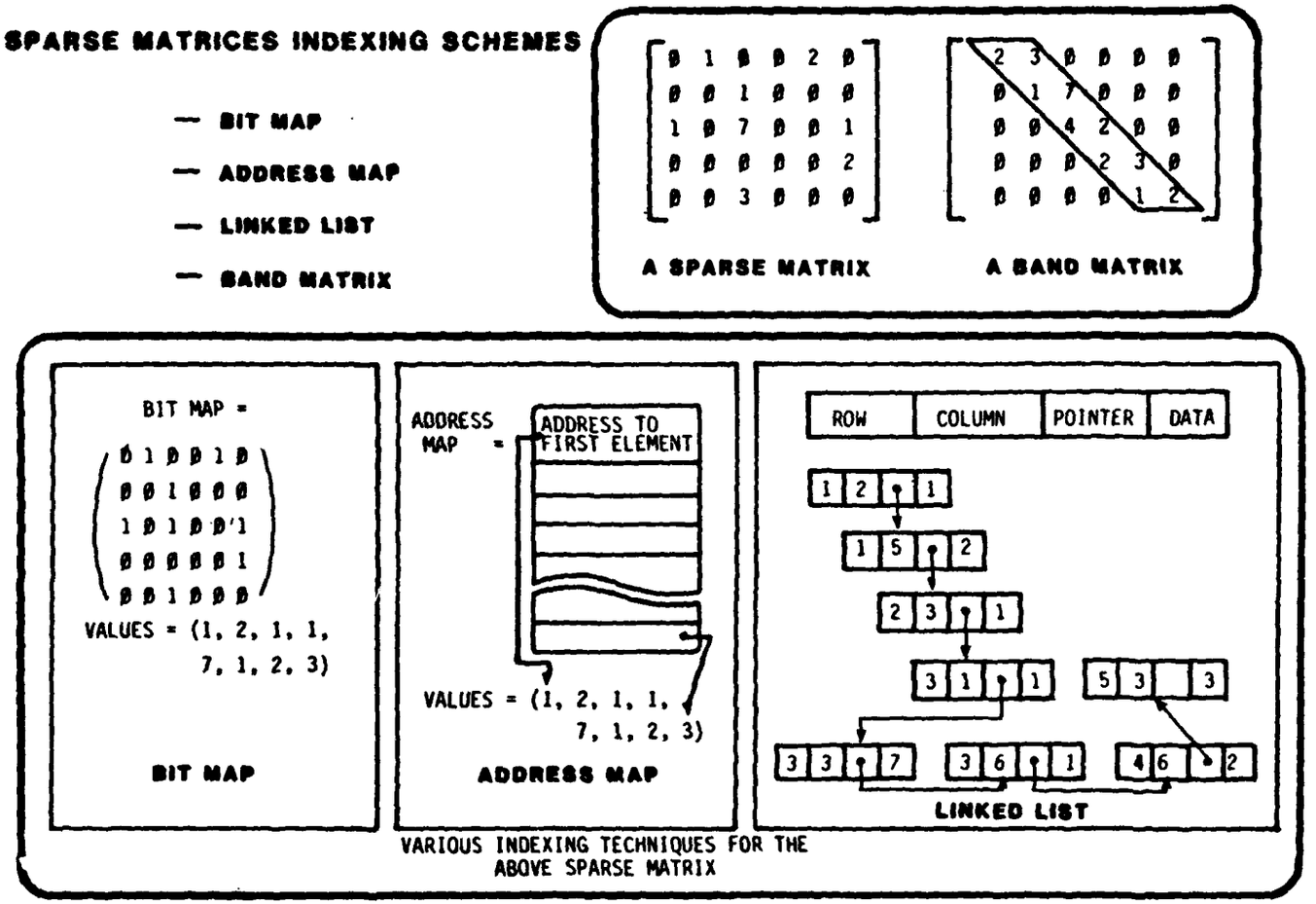


Figure 4-39. Sparse Matrices Indexing Schemes

Use of a single bit corresponding to a single element in the matrix is a bit map indexing scheme. The number of bits in the bit map is equal to the number of elements in the matrix. When a non-zero element occurs in the matrix, its corresponding position in the bit map is set to 1. Zero elements in the matrix will be identified as null entries in the bit map. The values for each non-zero element can then be stored in successive order in a separate list. To reconstruct the matrix the bit map is read until a bit set to one is encountered. This indicates a non-zero element for the matrix having the value of the next position in the list of values (Figure 4-39).

The address mapping scheme operates in a similar manner as the bit map technique. The only difference is that instead of setting a bit to one, an address or address displacement is recorded. Where the bit map requires only one bit for each matrix element, the address map uses considerably more storage.

Threaded or linked lists contain one record for each element of the matrix. The records hold the row, column, and value for each element. In addition a pointer to the next non-zero element is stored. An example of this scheme is shown in Figure 4-40.

A final sparse matrix indexing technique is used when the elements form a band or diagonal. Band matrices can be processed wholly or by rows and columns. When processing by rows or columns, matrix elements are stored in full vectors, one vector for each diagonal of the band matrix. Double indexing is avoided by creating an address table. Each address element holds the address of the first element of a corresponding vector.

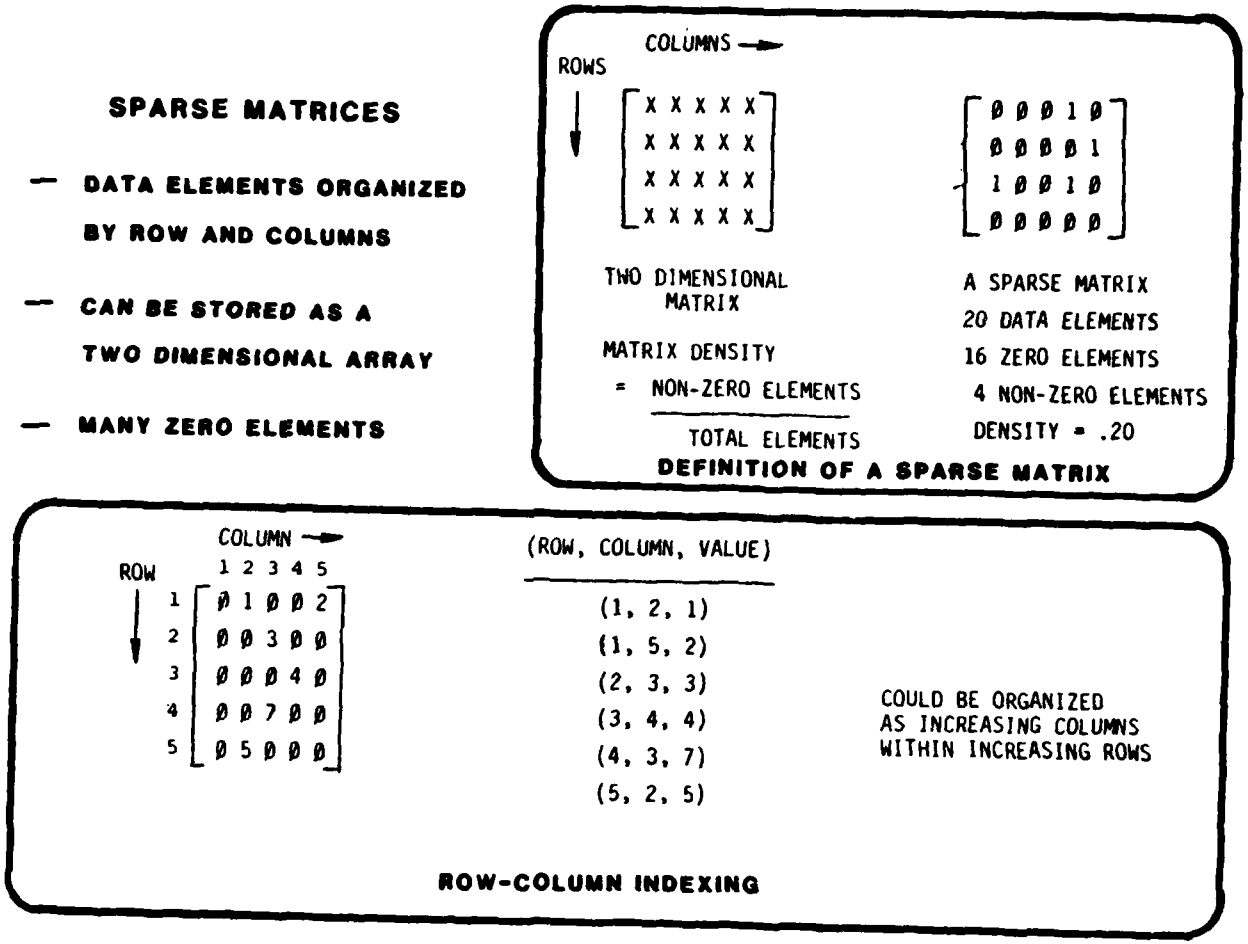


Figure 4-40. Sparse Matrices

SECTION 5. STUDY GOALS

The previous sections have given detailed overviews of the research performed on the IRDS study. Before delineating our conclusions and concepts which have arisen from this effort, we wish to refine the study goals. Section 5 presents a concise viewpoint of the target environment, the relevancy of raster topology, exploiting parallelism for raster data and the need for processing efficiency.

5.1 The Cartographic Environment

As we seek to more accurately record our rapidly changing environment, the cartographic production community will experience increased demands on its resources. The ultimate end users of cartographic products (i.e., battlefield commanders, planners, etc.), will expect timely updates to support their decision making. Concurrently, technology is abundantly supplying new and more precise source information. Thus, as the volume of information swells and our environment continues to change, the cartographic production environment will be required to accelerate its efforts.

A major component of the production process lies within the compilation and revision environment. Compilation here refers to the assembly and integration of a number of source materials for the purpose of producing an original cartographic product. Cartographic revision is the modification of a cartographic product due to changes in map purpose, geographic features or display format. Together these two functions represent the most critical work to be performed in the cartographic production process. It is also where a majority of human interaction is required and where the greatest benefits of raster technology can be realized. For these reasons our research on the IRDS study has been focused by the needs of the cartographic compilation and revision environment.

The potential for raster technology in the compilation/revision environment has been accepted. However, full scale implementation of all raster based systems has not. To fully understand the problems and seek out ways to resolve them it is first necessary to define the compilation/revision functions. Once the objectives and functions of a raster compilation and revision system are grasped, considerations for developing such a system must also be analyzed. Both of these areas are further defined in the following sections and provide the context of what the IRDS project has encompassed.

5.1.1 CARTOGRAPHIC FUNCTIONS FOR COMPILATION/REVISION

The manipulation of cartographic data is performed by a wide range of functions. In the compilation and revision environment, manipulation refers to performing some type of operation on digital cartographic data. With this definition, we can further differentiate functions which manipulate data into two separate categories. First there are functions which operate on the data while preserving its uniqueness. Display and logical functions exemplify this set of functions. A second and distinct group is composed of functions which institute some type of change to the physical data, such as an edit process would produce. Figure 5-1 depicts the hierarchy of the digital compilation/revision functions as defined by the IRDS study. Each of the two separate groups of functions are elaborated in the remainder of this section. The functions presented herein represent the tasks which must be implemented in a raster format in order to achieve an all raster environment for compilation and revision.

5.1.1.1 Functions Preserving Data Uniqueness

Manipulation of cartographic data is facilitated by automated tools which allow a user to better interpret the data prior to performing an actual compilation or revision task. Functions which support this capability are able to manipulate digital data without destroying the original information. In this sense, they serve to support the revision and editing processes.

Included in this group of functions are display oriented functions such as:

1. Displaying a set of features,

2. Enhancing the displayed information, and

3. Data transformations.

Displaying a set of features involves a user defined set of characteristics on which the system can retrieve features. This is used in conjunction with the extract feature function defined in a subsequent section.

Enhancing the display would include such things as modifying the color look-up tables, intensifying pixels and the designation of graphic symbolism. These functions might be used where a user wishes to define a different color set to adjust for personal differences. Other applications might require the use of color to identify possible discrepancies. Unions or intersections of features might appear in a contrasting color to allow the user to quickly locate areas of concern.

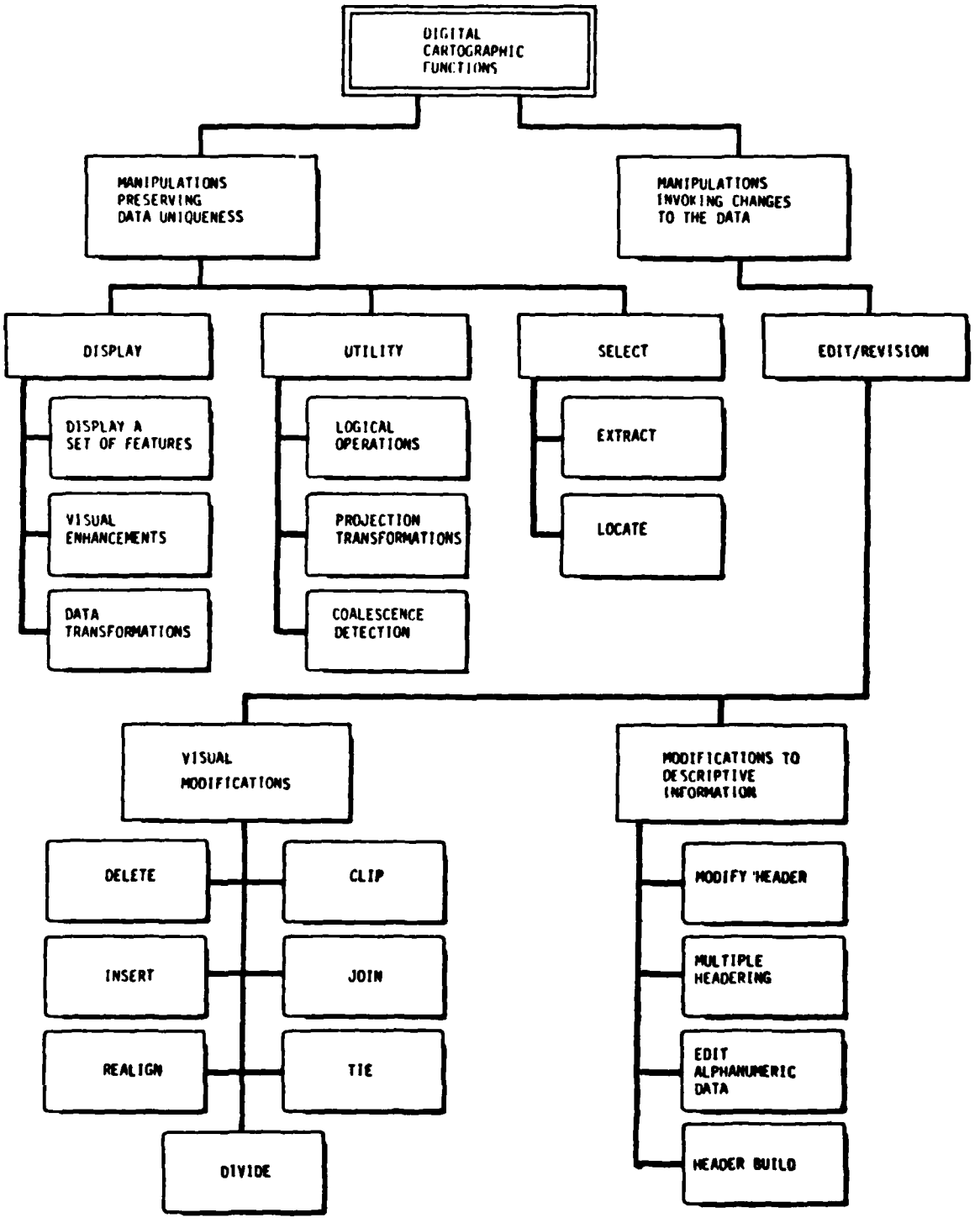


Figure 5-1. Digital Cartographic Compilation/Revision Functions

Data transformations include translation, rotation, and scaling of data. These functions permit users to view the data in different formats to gain further information or provide more rapid access to the data.

Utility functions also maintain data uniqueness and assist the user to gather information which might not readily be seen. This would include logical operations such as:

1. Inch counts

2. Area determinations

3. Orientation angles

4. Attribute statistics

Also included are projection transformations which allow the user to view the data in a different geographic reference frame. Coalescence detection would also be grouped here as it requires the system to process, detect, and output problem areas not easily seen by the user.

Another subset of functions preserving the original data can be referred to as select functions. Here two types of select functions are recognized as necessary for compilation and revision. First, an extract function would provide a global locate function which would enable the user to retrieve more than one feature at a time. The second type would permit the locating of a single feature for which further manipulation would be applied.

5.1.1.2 Functions Invoking Change

Compilation and revision processes necessitate modification to data in order to correct features to fit new purposes or situations. To carry out these tasks a set of functions which alter the data from its original state is required. Within this group of functions, two subsets can also be derived which change the visual characteristics or the descriptive information of features.

5.1.1.2.1 Visual modifications. One side of the compilation and revision functions deals with modifications to only the visual characteristics. The changes can easily be viewed by a user through either softcopy or hardcopy output. The specific functions relative to the compilation/revision environment are:

1. Delete - The elimination of a feature from the data and/or the display file is considered a deletion function. The deletion can either be logical or physical and impacts on the spatial characteristics and descriptive information concerning a feature.

2. Insert - This function allows for the creation of a new feature which is logically and physically added to the data file.

3. Clip - A reduction in the size or length of a feature is accomplished by a clipping function. Clips can be performed on a local or global basis as defined by a user with the clipped portion being deleted. The area to be clipped should be user definable, taking the form of straight or curved lines, or polygons. Examples of clipping are shown in Figure 5-2.

4. Divide - The divide function operates on the spatial characteristics of the data as well. A single feature can be divided into two or more separate entities as shown in Figure 5-2.

5. Join - The aggregation of features or feature segments into a single feature defines the join function, as shown in Figure 5-2.

6. Re-Align - A re-alignment of data involves some type of modification to a portion of a feature. Examples of this type of edit/revision function are given in Figure 5-3. This function is considered to be feature oriented in that it operates on a feature-by-feature basis.

7. Tie - Creating a relationship between physically unrelated features can be achieved by a tie function. In raster format, some data may visually appear and be interpreted as a complete feature such as the road in Figure 5-4. However the data may not be linked as a complete feature. The tie function makes it possible to manipulate separate features in unison without destroying their uniqueness.

5.1.1.2.2 Modifications to descriptive information. While change to the visual characteristics may be a primary requirement, modifying the narrative information will also be needed. Some of these functions may be invoked simultaneously with the visual modification tasks. For example, when a new feature is created, a header will also be required. The functions providing descriptive modifications include:

1. Modify Header

2. Multiple Headering

3. Header Build

4. Edit Alphanumeric and Point Symbol Information

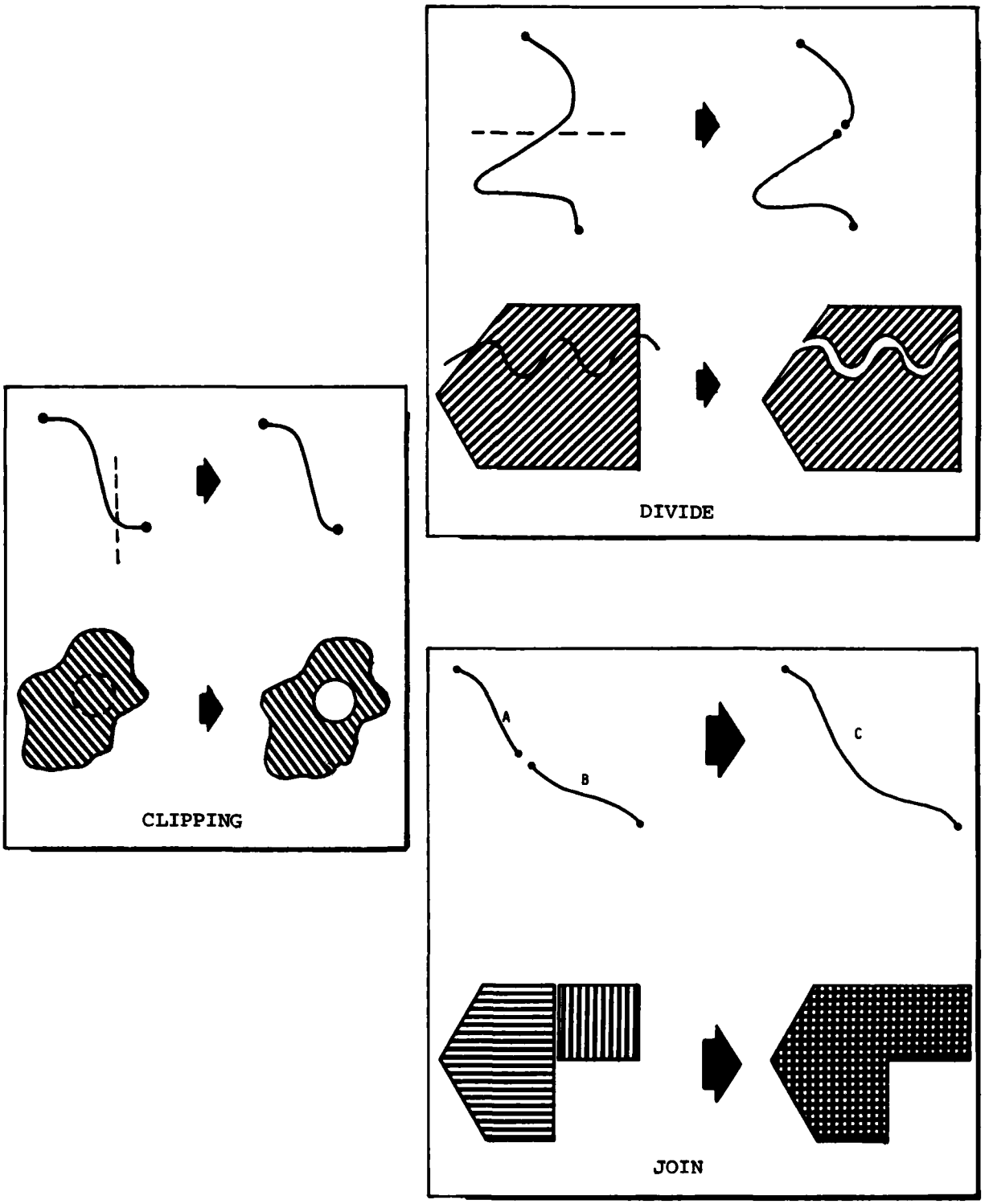
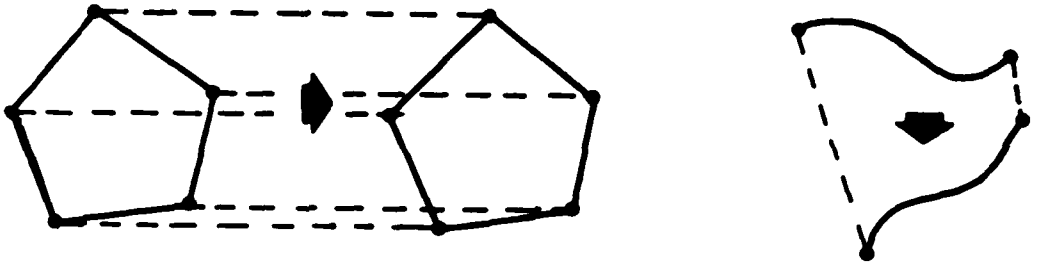
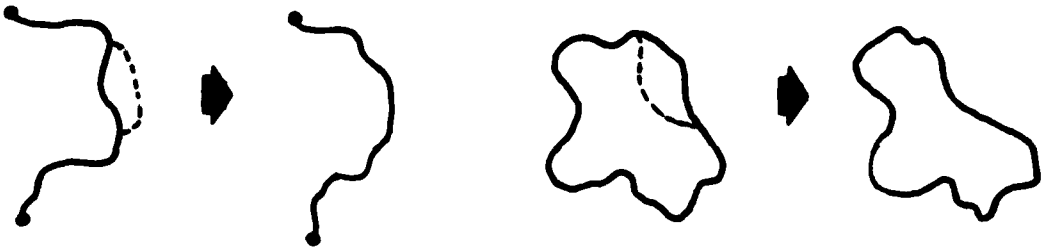


Figure 5-2. Visual Modifications

MOVE OR OFFSET A FEATURE
- ROTATE
- TRANSLATE
- SCALE



UPDATE INTERNAL CHARACTERISTICS



UPDATE ENDPOINT CHARACTERISTICS

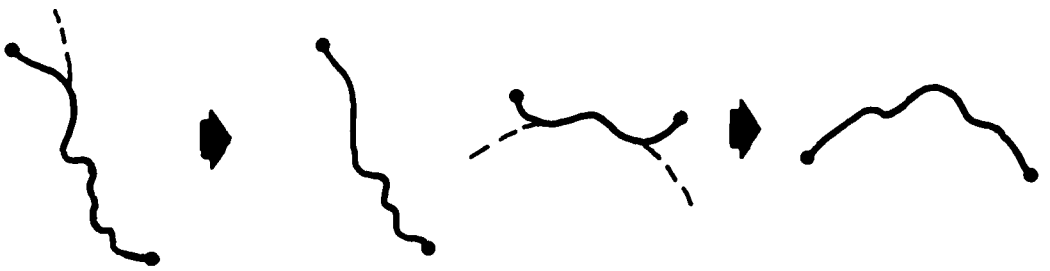


Figure 5-3. Realignment Examples

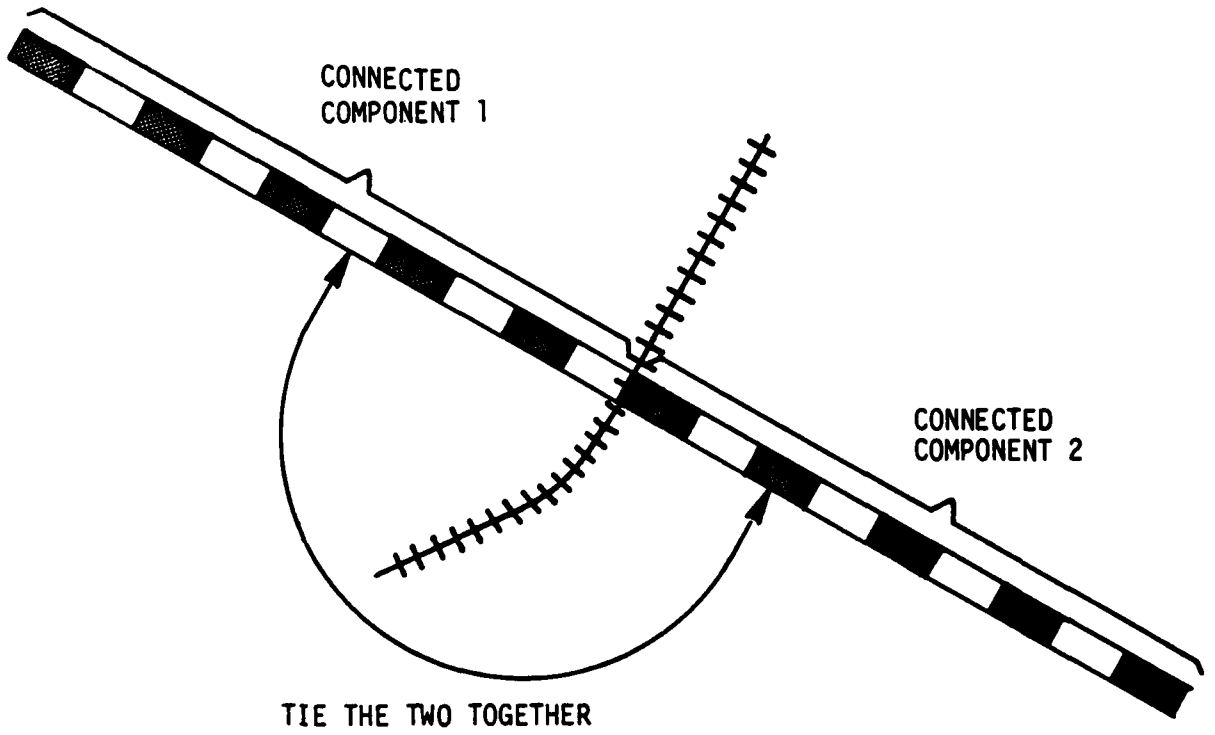


Figure 5-4. The Tie Function

Modifying a header includes the capability to change, add, or delete a features attributes and attribute values. Changes would be made to the data file as either logical or physical changes.

Multiple headering would supply the capability to associate a single feature to more than one header. This could be accomplished by using a pointer to an already existing header or the creation and addition of a new header.

Provisions must be made to allow a user to specify a unique header for each feature. Header build would provide this capability allowing for the creation of new headers. Feature attributes would be stored in the header along with other pertinent information such as feature I.D.

All cartographic products represent a collection of symbology and textual information. We use these elements to describe cartographic data in a meaningful way. In some situations, the location of symbology or the text may be of primary importance. Since we allow for the creation and placement of symbolic point and textual data, we must provide a function to also edit and revise them.

The functions which operate on alphanumeric text must be able to recognize a string of text as a feature to be modified and the individual characters to be modified. Editing and revision of symbology must also recognize the symbol along with its center position and possible orientation.

5.1.2 CONSIDERATIONS FOR A RASTER COMPILATION AND REVISION SYSTEM

Defining the desired functions provides a foundation from which to achieve the objectives of the IRDS study. Complementing these functional descriptions is the analysis of areas influencing the design, selection, and implementation of an appropriate solution for handling raster data. Primary considerations include:

1. Data Input,
2. The Human Interface,
3. Hardware, and
4. Future Requirements.

5.1.2.1 Data Input

The compilation and revision environment requires a system which will rapidly capture data, make it quickly available for manipulation purposes

and then output a final product. Thus the data to be processed will be temporal in nature. This necessitates a data structure which facilitates manipulation rather than storage.

Additionally, the variety of analog input sources may also impact on the data structure requirements. It is envisioned that a compilation/revision system would be capable of accepting imagery as well as existing cartographic manuscripts. Different information formats may call for distinct processing procedures and should be considered when designing a data structure which will be manipulated.

5.1.2.2 The Human Interface

The compilation/revision process is perhaps the most intense area of human interaction of the entire mapping process. Hence relevant considerations are extremely important. Functions should be designed and implemented which will not require more human directives than needed. The more steps involved, the greater the chances of error, fatigue, and rejection of the system.

An all raster compilation and revision system must also permit a user to think in vector terms while the system works in a raster mode. The natural conceptualization of digital data as features is unlikely to be reversed. Conditioning users to think in raster while performing manipulation functions is an extremely difficult task. Furthermore, it is undesirable as it places limitations on the data. Users must work at a smaller scale to effectively manipulate raster elements and the lower level of operations reduces the amount of visable data which aids the user in making decisions (i.e., conflict resolution).

Also to be considered is the systems' response time and the amount of data to be processed. Editing or manipulation of existing manuscripts only involves a portion of the total available data. The entire analog source is not required and processing all of the information would be inefficient. Thus, we need a data structure which will focus on manipulating smaller areas, reducing processing overhead, and increasing the system's efficiency, response time, and productivity.

Additionally, an all raster system should permit the unrestricted manipulation of data as it appears on the analog input sources as well as final products. Full symbolic compilation and revision is possible with all raster systems and should be kept in mind when developing a data structure for these functions.

5.1.2.3 Hardware

An additional consideration which impacts the selection of a raster data structure is the hardware which will perform the processing tasks. A data structure should be optimized to take advantage of current hardware trends. Decreasing memory costs and VLSI assembly techniques offer larger amounts of available memory for intelligent, display terminals. This local storage capacity can be utilized by a compact structuring technique. A second hardware trend is the integration of common software functions into hardware. Current implementations include hardware zooming and color look up tables for rapid display responses. However, there are several common cartographic functions which can be transferred into a hardware design. Finally, it has long been recognized that raster data is inherently suited for parallel operations, which should be facilitated by a raster data structure.

5.1.2.4 Future Requirements

Finally, consideration must be given to the future objectives of the compilation and revision community. The increasing use of scanned and/or digital imagery and scanned map/chart sources is best handled in a raster mode. Future requirements dictate high data volumes delivered in a rapid response environment. These requirements make the provision for all raster compilation and revision systems very timely. To meet these future goals, steps taken now must be flexible enough to ensure the achievement of these goals and absorb future requirements.

5.2 Raster Topology

Geographic information represents space as a continuum in which geographic entities can have interrelationships as well as structural and geometrical descriptions. Data structures which describe a system of interconnections among individual entities are topological structures. Topology is the study of properties which are invariant of continuous transformations. The objects in Figure 5-5 have certain properties which remain the same in both images, such as adjacency, connectivity, and the number of holes, edges, and vertices describing each object. A complete set of topologic properties are presented in Figure 5-6. The study of these properties is important for many related fields. For example, shape comparison can be assisted by computing Euler numbers of shapes independent of rotation, translation, and scaling. Additionally, data structures which contain the topologic properties of adjacency and neighbors among entities are useful for geographic information system or data base applications where query and data exploitation capabilities are desired. However, the focus of this study effort was upon the property of connectivity among pixels. A set of pixels is connected if any two pixels in the set can be joined by a path. A figure without disjointed parts which is joined by a path is termed a connected component. It is important to note that by focusing upon pixel connectivity we are primarily concerned with spatial characteristics which are inherent to the connected component.

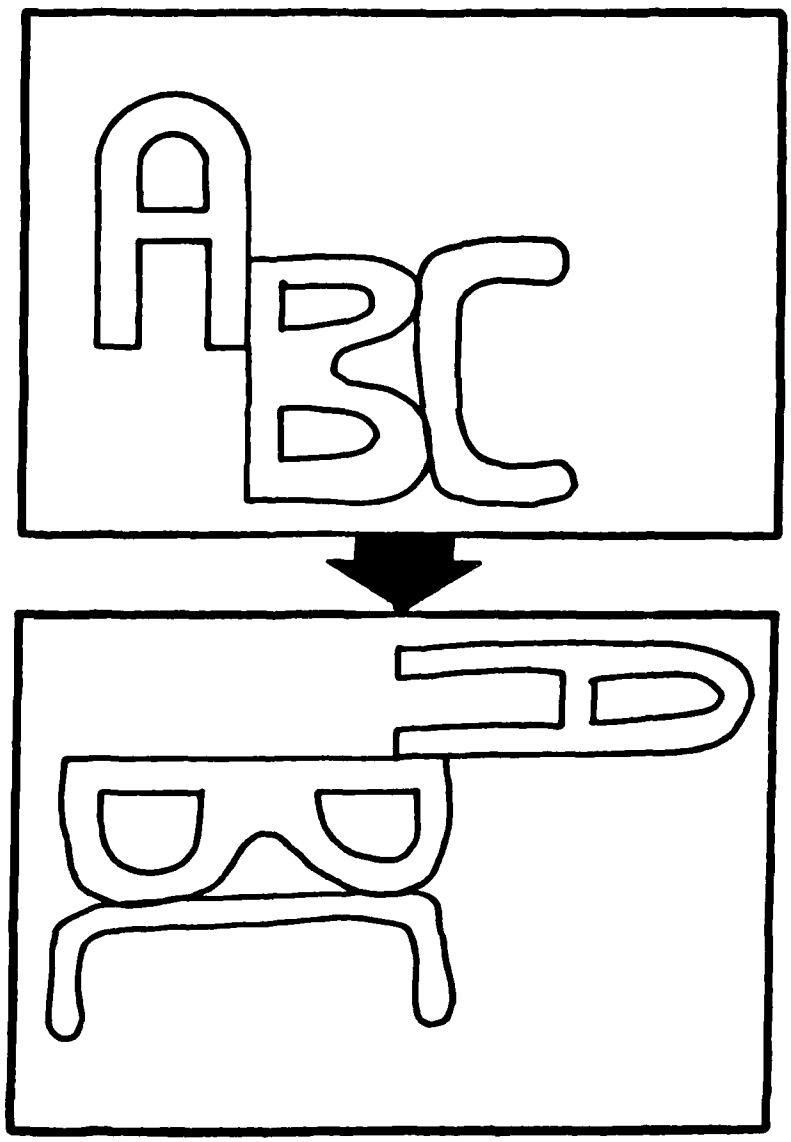


Figure 5-5. Spatial Transformation

- **FUNDAMENTAL CONCEPT**
  - **CONNECTEDNESS**
  - **RELATIONSHIPS AMONG SUBSETS**
- **ADJACENCY**
- **CONNECTIVITY**
- **SIMILARITY AND DIFFERENCES**
- **COMPONENTS**
  - **EDGES**
  - **HOLES**
  - **VERTICES OR NODES**
- **NEIGHBORS**
- **MAXIMUM AND MINIMUM POINTS**

Figure 5-6. Topologic Properties

Within the raster domain, the issue of connectivity can be subject to topological uncertainties. A clarification and resolution of these uncertainties is necessary before further definition of pixel connectivity.

5.2.1 TOPOLOGICAL DISCRETE IMAGES

True topological equivalency may not be possible in comparison between analog and discrete images. For example Figure 5-7 presents an ambiguous object which can be interpreted as one or four sets. Pavlidis (1982) introduced the definition of analog image and sampling grid compatibility which implies the preservation of topology. Pavlidis offers the following definitions to introduce the notion of topological compatibility between analog images and discrete images.

1. Sampling Resolution. The size of the sampling grid cell must be sufficiently small to preserve the shape of the region. If the length of the side of the sampling grid is h, then for every object there exists a circle of diameter $d > \sqrt{2h}$ that is tangent to a boundary and lies entirely within the object. Thus for shape preservation purposes a sampling resolution h can be chosen less than or equal to $d / \sqrt{2}$.

2. Preservation of Shape. Connected sets in the analog image must be topologically equivalent to connected sets in the discrete image. However, preservation of topology does not guarantee preservation of shape.

3. Topological Equivalence. Two sets are topologically equivalent if there is a one-to-one mapping between them and the mapping and its inverse are continuous. In the remainder of this report we will be concerned with the topological properties of discrete images.

5.2.2 CONNECTIVITY DEFINITIONS

The manipulation of connected components in raster images above the pixel level can be facilitated by a data structure that allows for the connectivity of pixels that constitute connected components. To enhance the discussion of this data structure, a few definitions are necessary.

1. Neighbors. Two pixels are direct neighbors (d-neighbors) if they each share a side, and are indirect neighbors (i-neighbors) if they are adjacent only at a corner (Figure 5-8).

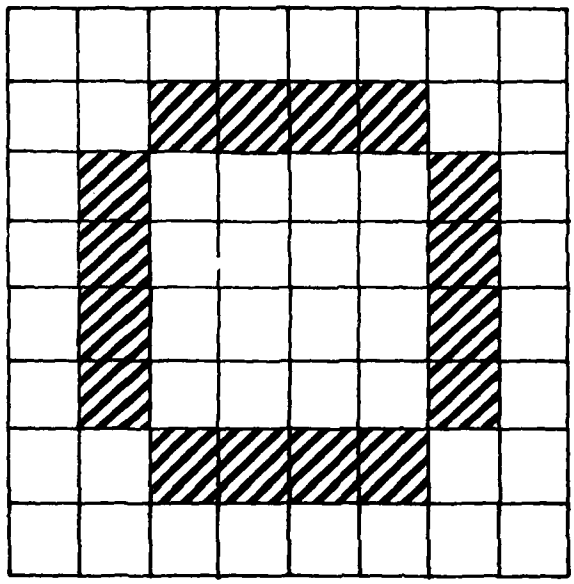
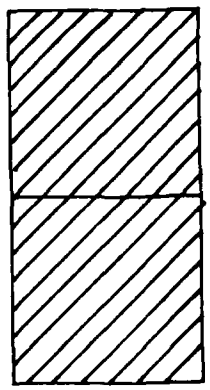
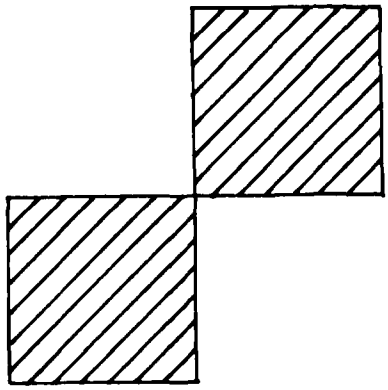


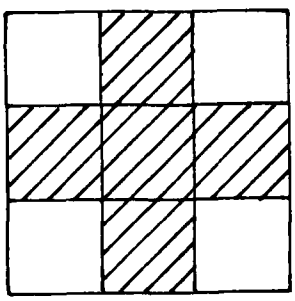
Figure 5-7. Ambiguous Object



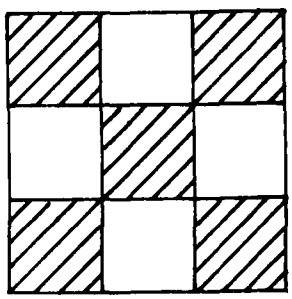
DIRECT
CONNECTIVITY



INDIRECT
CONNECTIVITY



DIRECTLY CONNECTED
NEIGHBORS



INDIRECTLY CONNECTED
NEIGHBORS

Figure 5-8. Definitions of Directly and Indirectly Connected Neighbors

An N-neighbor denotes the pixel whose position is marked with N as depicted in Figure 5-9. There are many existing algorithms that utilize this neighborhood numbering convention. D-neighbors are those which have an even N, i-neighbors are those with N being odd.

2. Connectivity. A set of pixels belonging to property set S are connected if for every pair of pixels in S, there is a path between them. A path is a sequence of pixels such that pi-1 is a neighbor of pi and pi+1 is a neighbor of pi. The usual convention for bilevel images is to use i-connectivity (corner touching allowed) for the black intensity and d-connectivity (only side touching) for the white intensity. This convention can be extended to color images in which the foreground intensity is i-connected and the background intensity (all other colors) is d-connected. The d-connectivity requirement ensures no ambiguities can exist.

3. Simple Path. A path such that no pixel has more than two d-neighbors in the path. A closed path is where the first and last pixel are identical.

4. Contour (i-contour) of a connected set R is the set of all pixels in R, which have at least one d-neighbor not in R.

5. Interior of a connected set R is the set of all pixels in whose d-neighbors are also in R.

6. Line is a set of pixels such all belong to the contour of the set. A line is a non-empty set that has an empty set of interior pixels.

7. Region is a set of pixels such that its subset of interior pixels is non-empty. A full region has more than four pixels, its contour is a simple path, and its interior pixel subset is d-connected.

5.3 Parallelism

Raster processing in the cartographic environment can be partitioned into four categories:

1. Data Base Management - such as storage, retrieval, and updating of raster files.

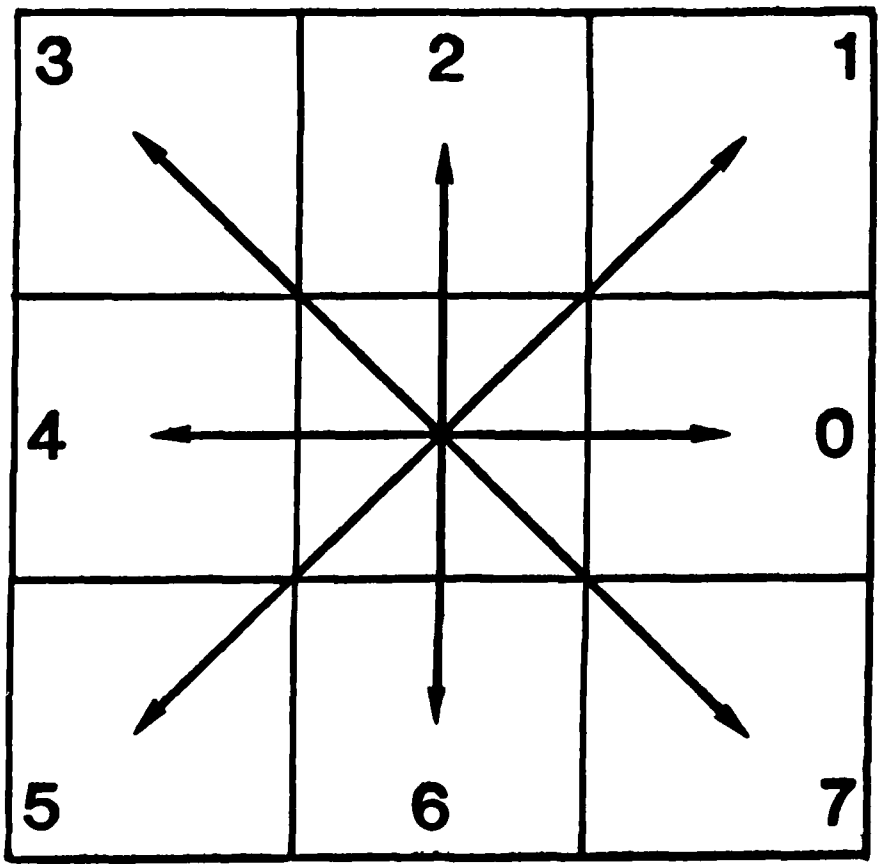


Figure 5-9. N-Neighbor Numbering Convention in a 3 X 3 Neighborhood

2. Display - such as scaling, rotation, and translation.

3. Encoding - such as component labeling and modification of attribute values.

4. Analysis - including filtering.

Each of these functional categories consist of both high level and low level operations. High level operations perform interpretation, are evaluative, and must be concerned with the control of the low level operations. This requirement restricts high level operations to sequential search techniques and data independent execution. Low level operations are tasked to feed information to the higher operations. Sample low level operations include masking, logical comparisons, neighborhood operations, noise suppression, edge detection, pixel counts and grey scale transformations. These simple functions account for a large amount of the overall processing burden. This processing load can be reduced by exploiting the topology which is inherent in the raster image and the simple computations which are applicable to the image. The low level operations process large quantities of highly structured data, thus optimization by parallelism and single-instruction multiple-data (SIMD) stream processing can greatly impact system response.

In order to facilitate the suitability of parallel processing techniques, it was a study goal to incorporate "activity bits" into a raster data structure to indicate those pixels which should be subject to a given parallel operation. Such a data structure greatly simplifies and enhances the development of parallel processing algorithms.

5.4 Improving Processing Efficiency

The inherent problems of raster data has required rigorous procedures to effectively manipulate it. Past manipulation techniques have relied heavily on user-machine communications to bring about desired results. However, as more work is required of the user, more time will be consumed in completing tasks. Additionally, the chances of error are greater and the full potential of the machine is ignored. This necessitates a data structure not only to facilitate manipulation, but also minimize the amount of work to be performed.

Providing a level of function intelligence is one alternative for improving processing efficiency. A system which can make automatic assumptions rather than waiting for user directives could greatly reduce the amount of required interaction. This in turn would speed up system throughput as processors would be idle less frequently.

For example, use of the join function implies that at least two features will be operated on. Therefore once the first feature has been identified, the system should automatically prompt the user for a second feature. When the second feature is given the system should be capable of determining the endpoints to join from the orientation of the two features. From this information the two features could be logically and visually joined. The user would be given the option of overriding the systems decision and manually create the join.

A second goal is to reduce the amount of processing levied on the system. By restricting processing to only necessary tasks, higher efficiency can be achieved. Scanning only the data to be edited rather than the entire analog source is one example. Processing only the segment of a feature affected by a manipulation as opposed to the entire feature is another.

Developing or acquiring the techniques to perform such tasks would help to minimize the amount of necessary work. We are not looking for the absolute answer. Yet by exposing issues such as function intelligence, concepts for the future all raster systems can be developed which will provide more efficient processing of raster data.

SECTION 6. DATA STRUCTURE ELEMENTS

In order to support the full range of cartographic manipulations, which were identified and described in Section 5.1.1, it is imperative that a raster data structure include three essential elements; a pixel value, a connected component label, and a pixel connectivity map. The following section will discuss techniques available for calculating these elements and place their creation and use in an overall processing flow. This section will briefly define each of these elements and describe the properties of the element.

6.1 Pixel Value

Raster scanned images consist of a sample light intensity function, i(x), over some source material. This intensity function can be either a black and white scalar function, yielding a grey scale image, or a color image function. Color image functions are a combination of red, green, and blue (RGB) light intensity functions which can be treated as a vector with red, green, and blue components or as three coincident scalar planar images. When the scanned source material is a multicolor map or chart, the colors can be reduced to a small subset reflecting a range of RGB percentages or hue, intensity, and saturation combinations (HIS).

Hue indicates the relative amount of a primary color; intensity dictates the brightness of the color and saturation indicates the amount of white which dilutes the color. In either case, the three properties RGB or HIS, can be quantified and represented by some value where 2^N equals the number of bits available. Using an eight bit word to store a color value permits the display of 2^8 or 256 distinct colors. Due to permissible variations of the three color components, the total of colors from which the 256 can be chosen is $2^{8\ (3)} + 2^{24}$ or 16.7 million colors.

The 8-bit pixel value is the current common data value used in input, display, manipulation, and output for raster images.

Manipulations which would commonly be performed upon the pixel value would include indicating a class of interest. By specifying a single value, a range of values or several separate values of an entire image would be reduced to essentially a binary image; those pixels of interest (object) and those pixels not of interest (background) (Figure 6-1). Although the full range of pixel values would continue to be displayed, specifying

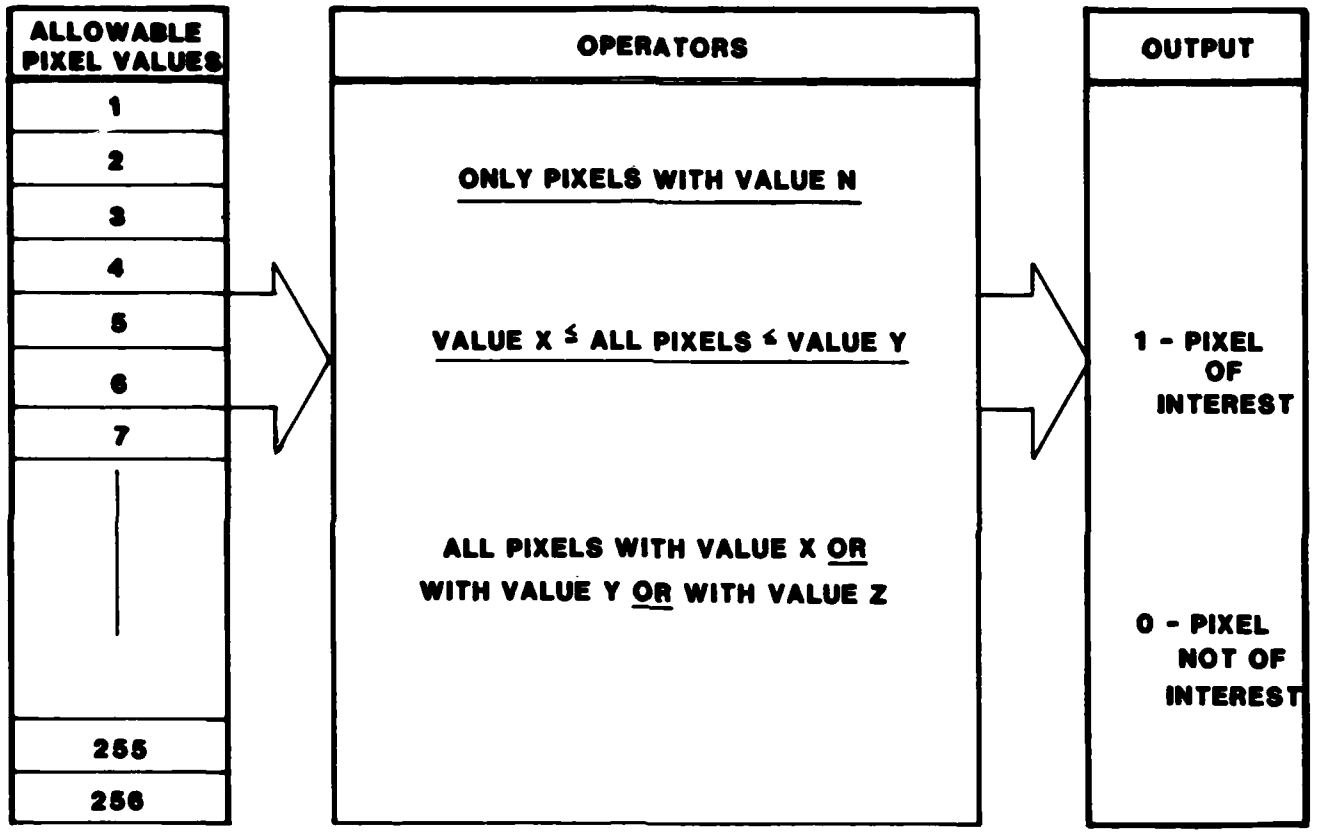


Figure 6-1. Indicating a Class of Interest

a class of interest creates a fundamental distinction for the purposes of further processing.

6.2 Pixel Connectivity Map

Many raster processing algorithms depend upon locally searching and computing functions for the 3x3 neighborhood containing the eight neighbors of a pixel. Processing efficiency can be realized by a data structure which encodes connectivity within the neighborhood above the pixel level. The notion of encoding the 3x3 neighborhood connectivity in a compact 8-bit code was introduced by Sobel (1978) to assist in extracting image contours. Lougheed et al (1980) uses a similar structure to facilitate 2-D transformations.

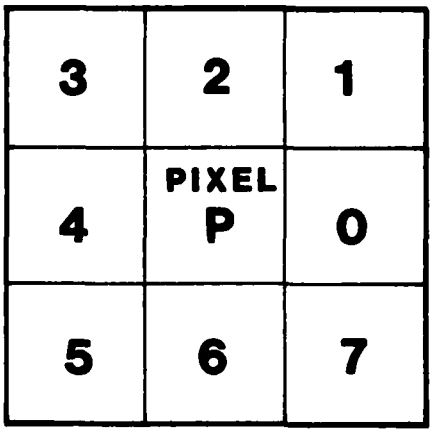
The purpose of this encoding is to reduce the computational burden of accessing and processing local neighborhoods. Use of an 8-bit pixel connectivity map (PCM) subdivides the computational burden into a single accessing of the eight neighbors, assembling them into the PCM and then computing a range of neighborhood functions by simple lookup and masking operations on the PCM.

The pixel connectivity map consists of an 8-bit field (one byte) in which each bit number corresponds to the eight possible neighborhood positions (Figure 6-2). For pixels in the 3x3 neighborhood which are of the same class as the center pixel would have their corresponding bits set. Figure 6-3 illustrates a 3x3 neighborhood in a bilevel image and shows the corresponding pixel connectivity map for the center pixel.

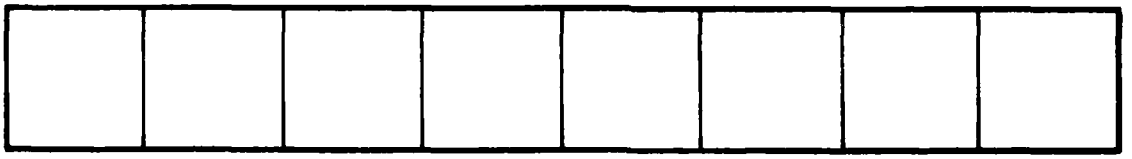
6.2.1 PROPERTIES OF THE PIXEL CONNECTIVITY MAP

The pixel connectivity map facilitates parallel operations by enhancing pixel searching and accessing, but the real power of the PCM comes from the ability to interrogate a PCM to uniquely classify an individual pixel. These unique classifications allow us to determine the important characteristics of a connected component which are referenced by the cartographic manipulations. Simple logical comparison operations (AND, OR, NOT) apply a set of predefined bit mask templates against each PCM of a connected component to detect:

1. Interior pixels,
2. Contour pixels,
3. Arc endpoints,
4. Local minima and maxima

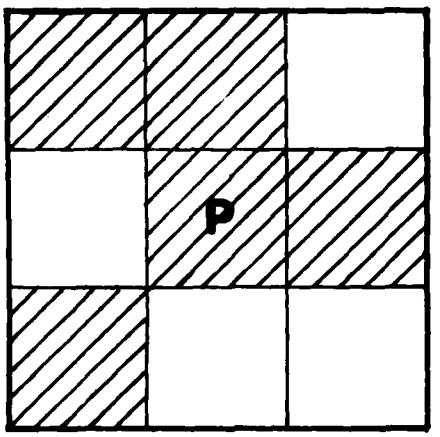


3x3 Neighborhood For Pixel P



Pixel Connectivity Map For Pixel P

Figure 6-2. Encoding 3 X 3 Neighborhood in a Pixel Connectivity Map



3x3 Neighborhood Of Pixel P

0	0	1	0	1	1	0	1
BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0

Pixel Connectivity Map For Pixel P

Figure 6-3. Example of Pixel Connectivity Map

5. Minima and maxima of holes within a connected component,

6. Junctions or intersections between connected components.

Detecting an interior pixel is a simple operation which tests the pixel connectivity map for all bits representing directly connected neighbors or d-bits (0,2,4,6) being set to 1. We can 'AND' a bit mask template (01010101) against each PCM and for each result equal to the template we have an interior pixel and for each result not equal to the template we have a contour pixel. This operation is depicted graphically in Figure 6-4. Detecting interior pixels of a connected component separates pixels into two mutually exclusive subsets; interior pixels and contour pixels. Every pixel of a connected component is either an interior pixel, or a contour pixel of which there are several special classes.

Contour pixels are identified by possessing at least one directly connected neighbor not a member of the connected component and at least one directly connected that is a member of the component. Thus the PCM contains one of the d-bits set to zero and one of the d-bits set to one as shown in Figure 6-5.

A special class of contour pixels are lines of a single pixel width. These pixels are a subset which have opposite neighbors, either i-neighbors or d-neighbors, as members of the component and all other neighbors excluded. Thus the PCM has d-bit combinations of 0 and 4, 1 and 5, 2 and 6, or 3 and 7 set to one. Figure 6-6 illustrates the complete set of single pixel width lines in a 3x3 neighborhood and the bit mask templates for detecting those lines.

Arc endpoints in a connected component are a class of contour pixels with only one neighbor. In Figure 6-7, pixel 2,3 is recognized by its distinctive PCM (01000000), whereby only one neighbor is indicated. A collapsing arc is also easily recognized by having only contour pixels as having more than one neighbor, all of which are contour pixels.

Cederberg (1980) used a 3x3 neighborhood template to identify the local minima and maxima pixels of a connected component in the generation of the Raster Scan Chain Code data structure. A complete set of templates for detecting minima and maxima pixels in external contours and arcs is given in Table 6-1. External contour endpoints are shown graphically in Figure 6-8. The templates which are used for detecting the local minima are obtained by a 180° rotation of the local maxima templates. This rotation is equivalent to a four place logical rotation (shift with wrap-around) of an 8-bit register, as shown in Figure 6-9. This is significant for reasons of space efficiency and for use in those operations where performing a logical rotation is faster than accessing a new template.

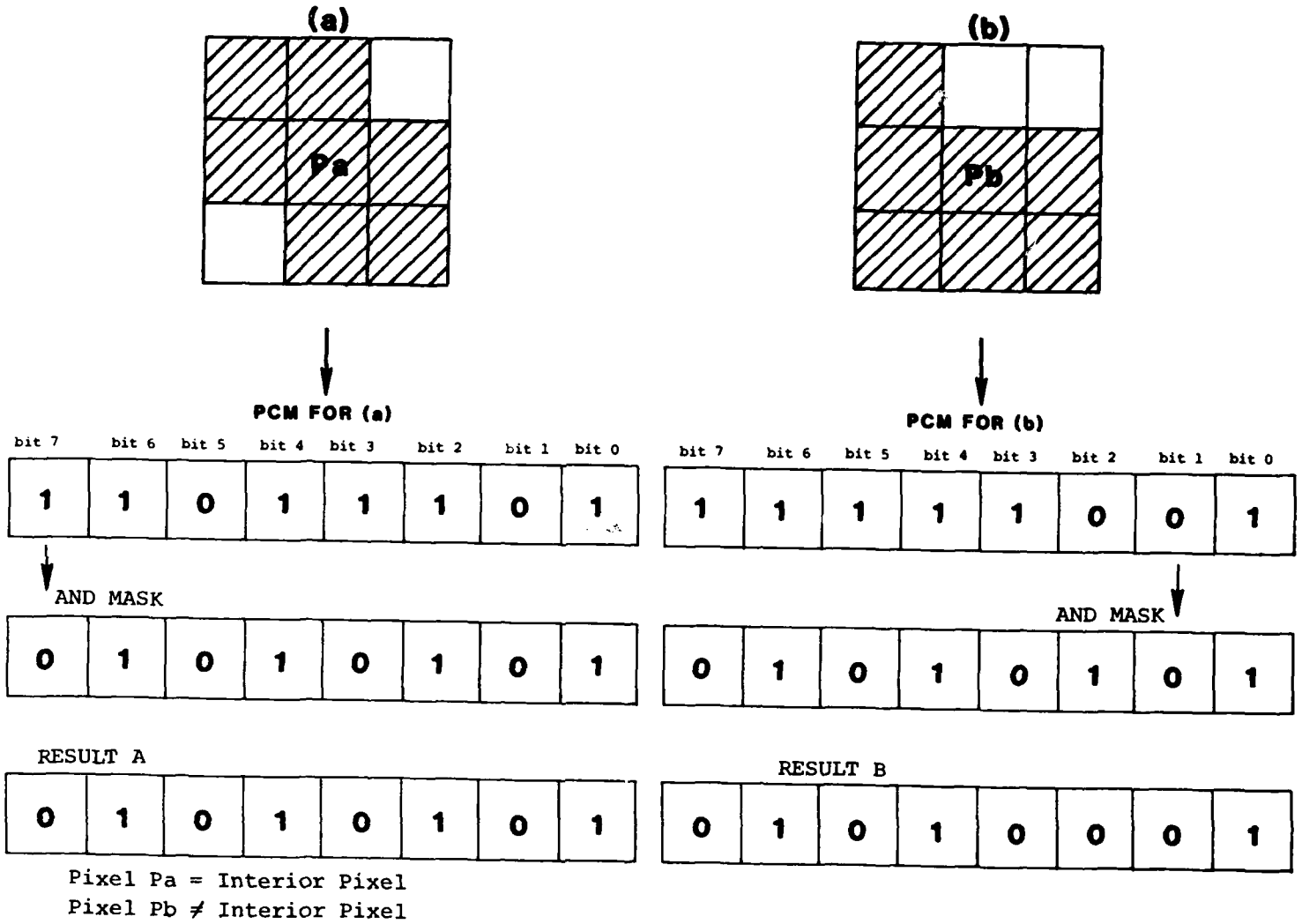
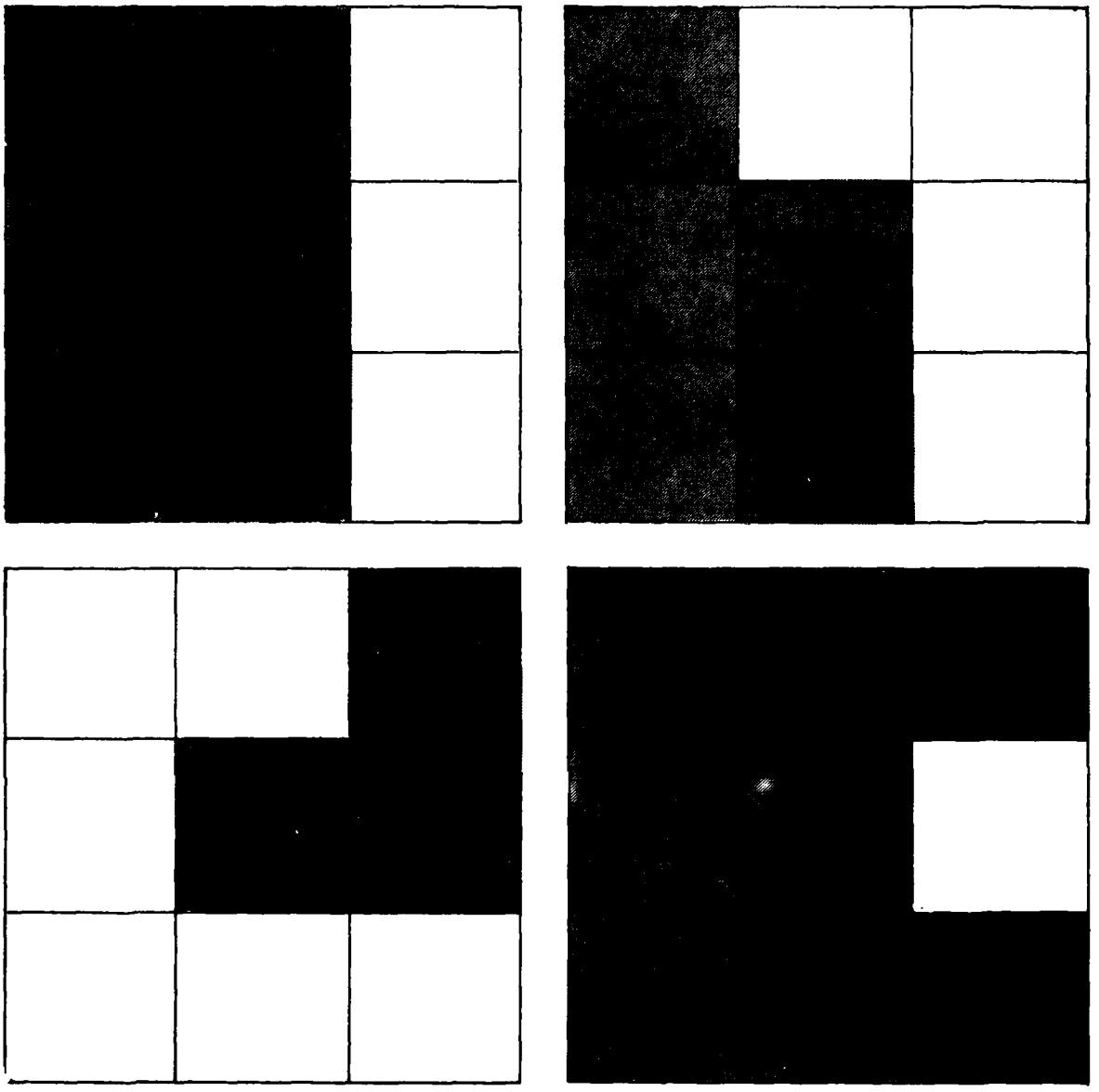
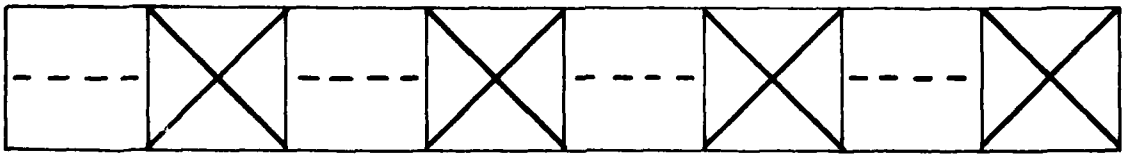


Figure 6-4. Detecting Interior Pixels

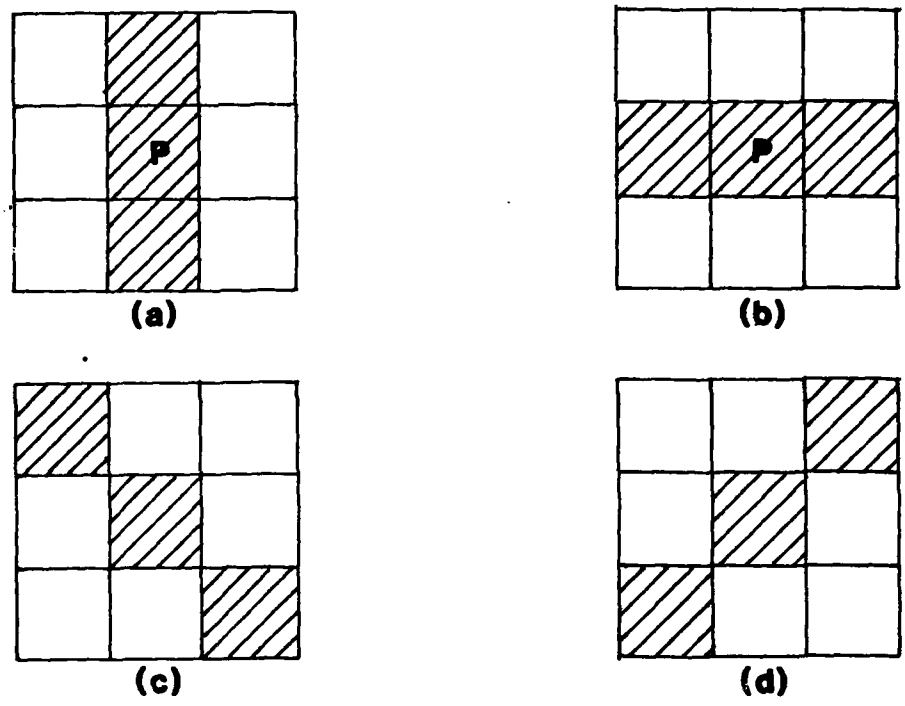


Examples of Contour Pixels in a 3x3 Neighborhood



PCM for a Contour Pixel

Figure 6-5. Detecting Contour Pixels



Single Pixel Width Lines in a 3 X 3 Neighborhood

(a)	0	1	0	0	0	1	0	0
(b)	0	0	0	1	0	0	0	1
(c)	1	0	0	0	1	0	0	0
(d)	0	0	1	0	0	0	1	0

Bit Mask Templates for Detecting Single Pixel Width Lines

Figure 6-6. Single Pixel Width Lines

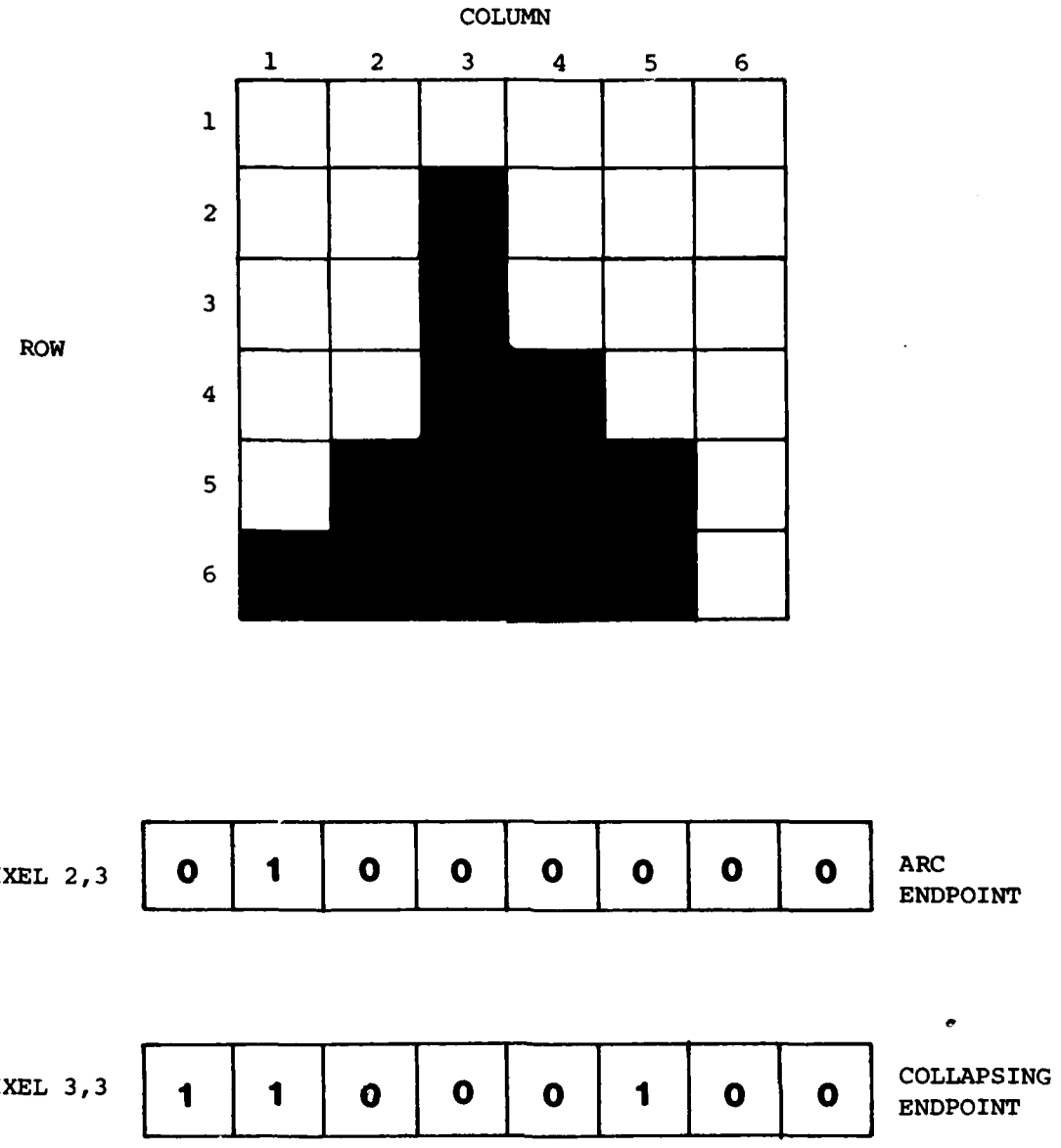


Figure 6-7. Arc Endpoints and Collapsing Arcs

Table 6-1.

Minima and Maxima Templates for External Contours and Arc Endpoints

NEIGHBORHOOD		PIXEL CONNECTIVITY MAP
000 011 1XX	=	XX100001
000 010 1X1	=	1X100000
000 011 01X	=	X1000001
000 010 110	=	01100000
000 010 011	=	11000000
000 011 001	=	10000001

External Contour Maxima Templates

NEIGHBORHOOD		PIXEL CONNECTIVITY MAP
XX1 110 000	=	0001XX10
1X1 010 000	=	00001X10
X10 110 000	=	0001X100
011 010 000	=	00000110
110 010 000	=	00001100
100 110 000	=	00001100

External Contour Minima Templates

NEIGHBORHOOD		PIXEL CONNECTIVITY MAP
000 010 100	=	00100000
000 010 010	=	01000000
000 010 001	=	10000000
000 011 000	=	00000001

Arc Endpoint Maxima Templates

NEIGHBORHOOD		PIXEL CONNECTIVITY MAP
001 010 000	=	00000010
010 010 000	=	00000100
100 010 000	=	00001000
000 110 000	=	00010000

Arc Endpoint Minima Templates

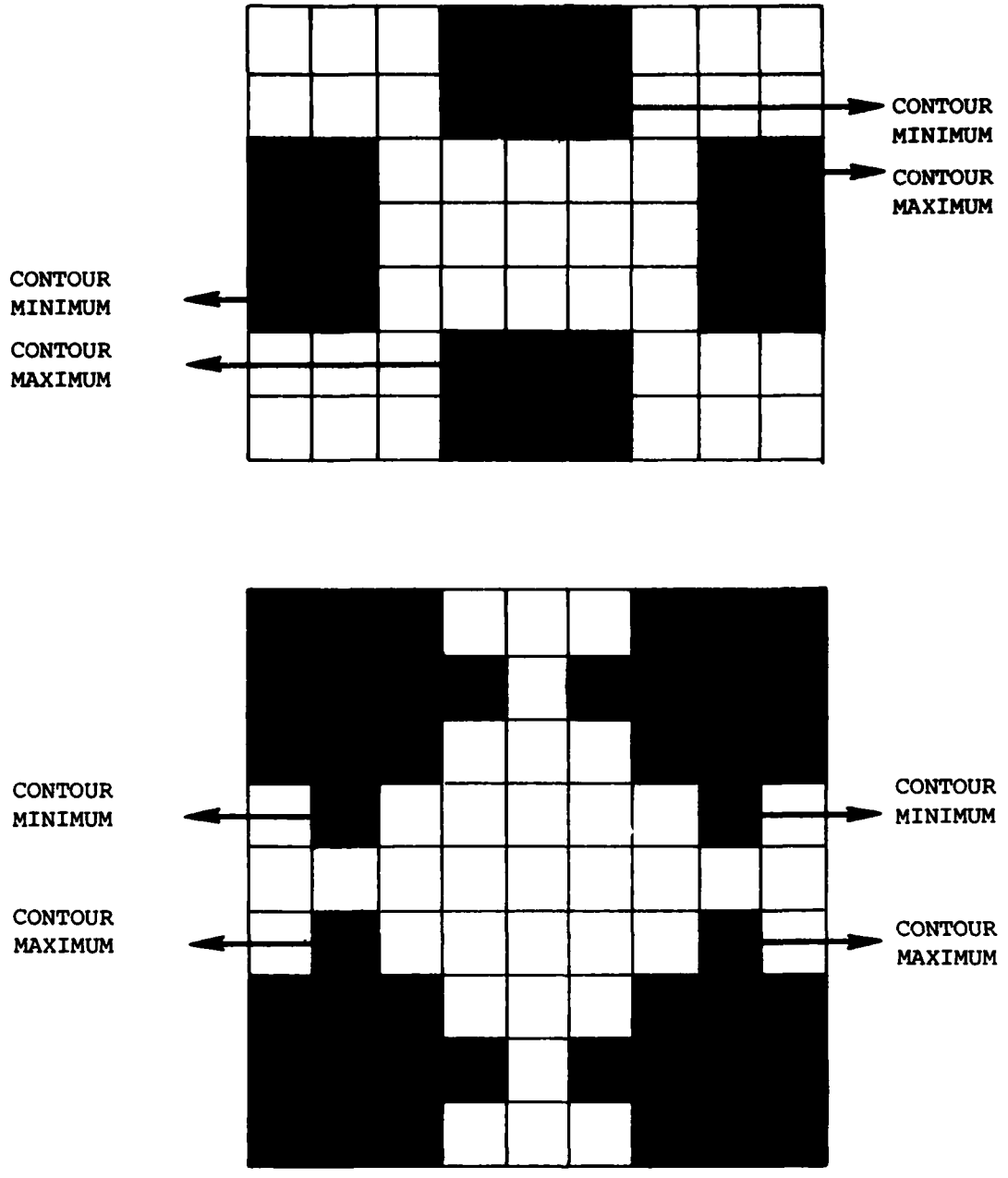


Figure 6-8. External Contour Endpoints

NEIGHBORHOOD

3	2	1
4	P	0
5	6	7

PIXEL CONNECTIVITY MAP

= 76543210

7	6	5
0	P	4
1	2	3

= 32107654

ROTATED NEIGHBORHOOD
(180°)

PIXEL CONNECTIVITY MAP AFTER 4
PLACE LOGICAL ROTATE

Figure 6-9. Minima/Maxima Neighborhood Template Rotation

The local minima and maxima of holes or object-background transitions can be detected by the templates presented in Table 6-2. Examples of hole maxima and hole minima pixels are provided in Figure 6-10.

By applying the templates for detecting hole minima and hole maxima to other areas of an image, midpoints of component junctions or intersections can often be detected. Junctions of degree four or x-junctions can be detected by the occurrence of hole minima and hole maxima within close proximity to one another usually a distance not greater than the width of the intersecting component. Figure 6-11, illustrates this principle for single pixel width lines and shows that the hole minima and hole maxima points can even coincide (6-11a and 6-11b). For junctions between thicker line segments it is possible to compute the intersection of the x-junction as the midpoint between the hole minima and hole maxima (Figure 6-12). Component junctions of degree 3 or T-junctions are typically detected by the presence of solitary hole minima and hole maxima (Figure 6-13). In this case the intersection midpoint can only be approximated by arbitrarily assigning an intersecting line along one of the sides of the detected hole minima or hole maxima. Unfortunately there are also some ambiguous situations where junctions may be indicated within a single connection component, as illustrated by Figure 6-14. Although these cases are not junctions as previously defined, they may serve to indicate breakpoints when finer division of features is required.

6.3 Pixel Label

Given an array of pixels, a fundamental operation is the aggregation and identification of pixel sets based upon a physical relationship. A pixel label is a unique identifier which is assigned to all members of a connected set. It is assumed that only those pixels which belong to a class of interest would be assigned labels and all background pixels would remain unlabeled.

Labels can be assigned according to the amount of aggregation which is necessary to support the desired cartographic manipulation at the desired level of resolution. The three levels of labeling offer finer resolution as:

1. Fight-connected components; both directly and indirectly connected as in Figures 6-15 and 6-16.

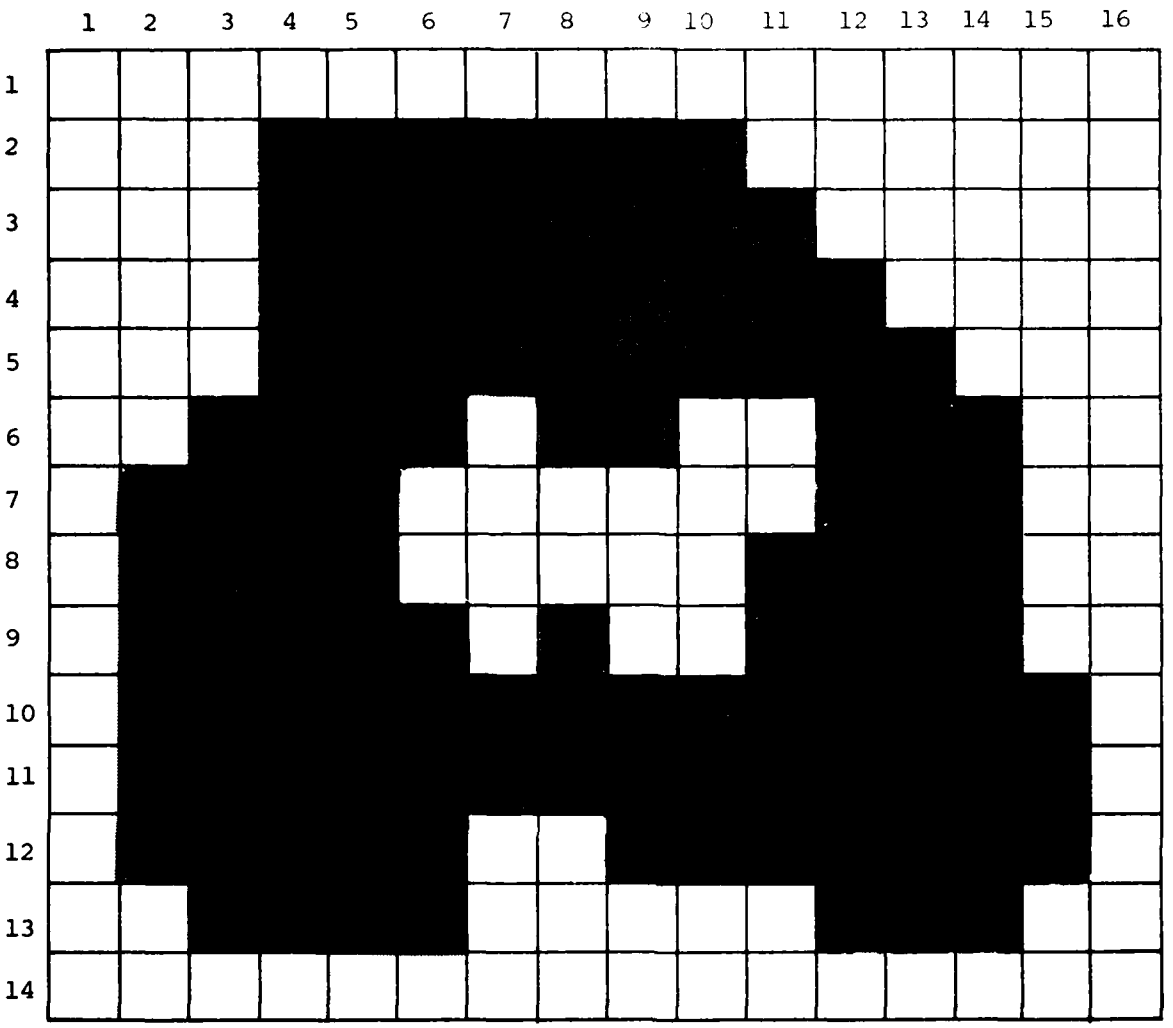
2. Eight-connected components spatially segmented at X and T junctions to create separately labeled segments.

3. Directly connected components as in Figure 6-15.

Table 6-2

Templates for Object-Background (Holes) Transitions

NEIGHBORHOOD	PIXEL CONNECTIVITY MAP	TYPE
XXX X11 100	= 001XXXX1	HOLE MAXIMA
XXX X1X 101	= 101XXXX	HOLE MAXIMA
001 11X XXX	= XXX1001X	HOLE MINIMA
101 X1X XXX	= XXX101X	HOLE MINIMA



PIXEL	PIXEL CONNECTIVITY MAP	CLASSIFICATION
A	10111111	Hole Maxima
B	00111111	Hole Maxima
C	11110011	Hole Minima
D	11111011	Hole Minima

Figure 6-10. Examples of Hole Maxima and Hole Minima Pixels

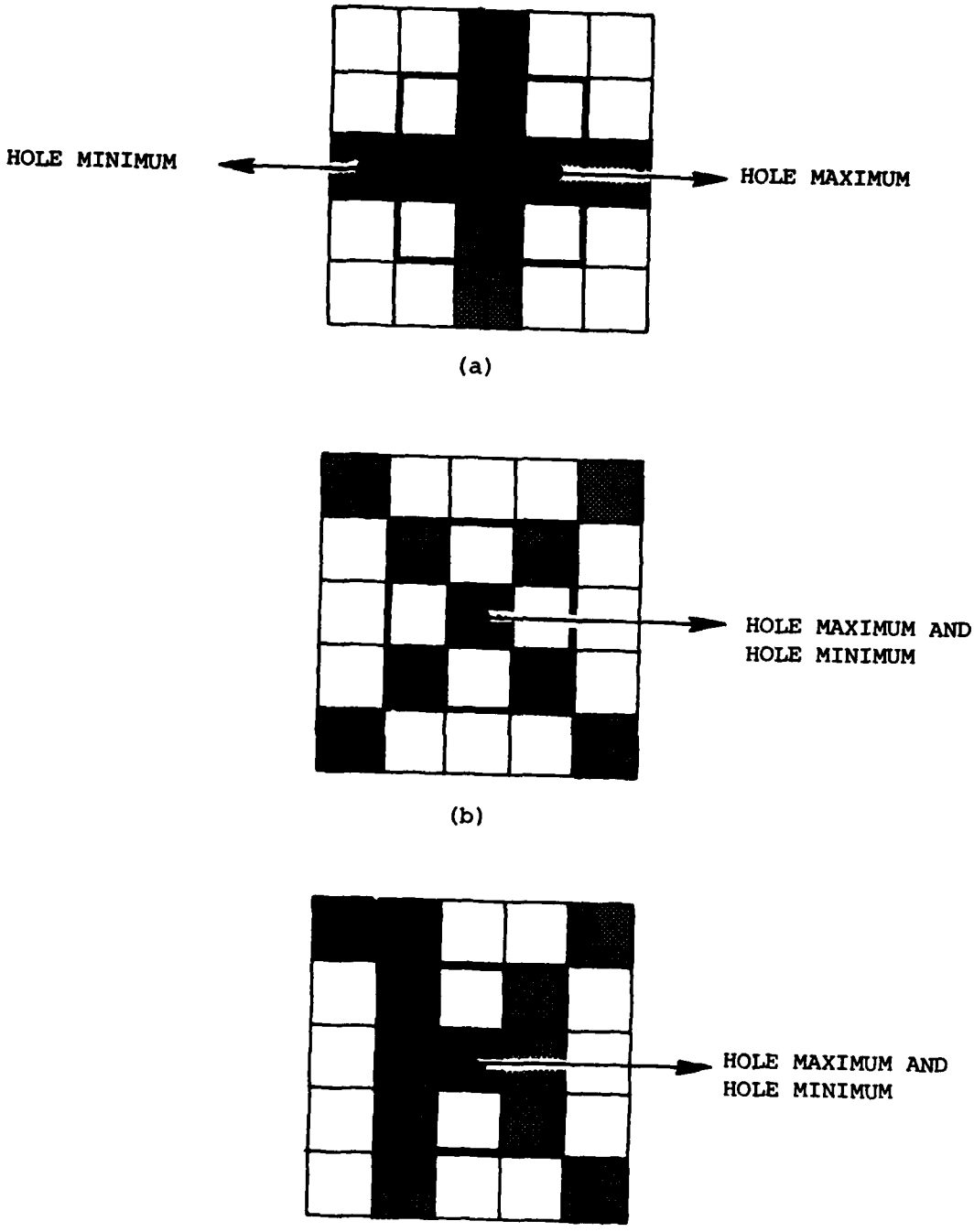


Figure 6-11. Single Pixel Width Lines With Junctions of Degree 4

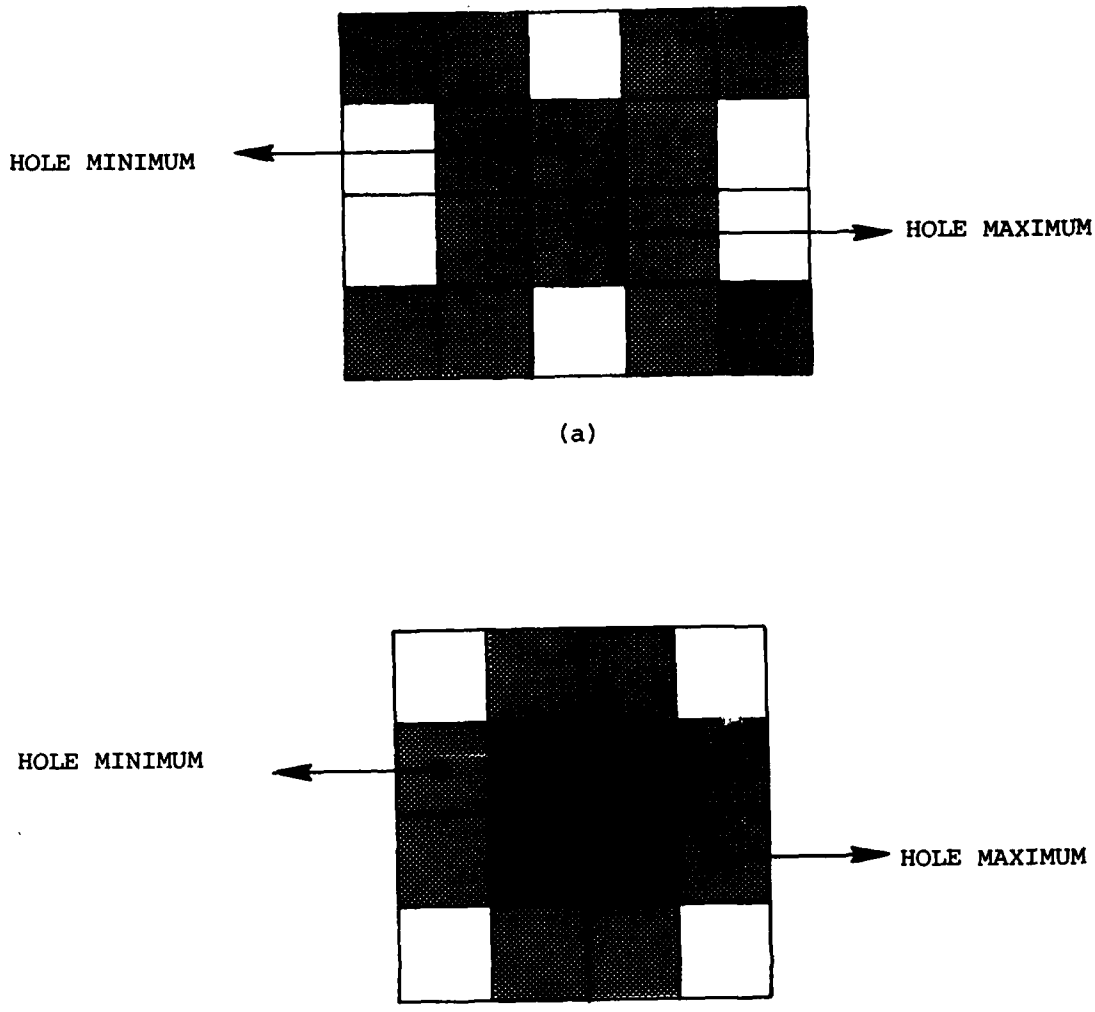


Figure 6-12. Multiple Pixel Width Lines With Junctions of Degree 4

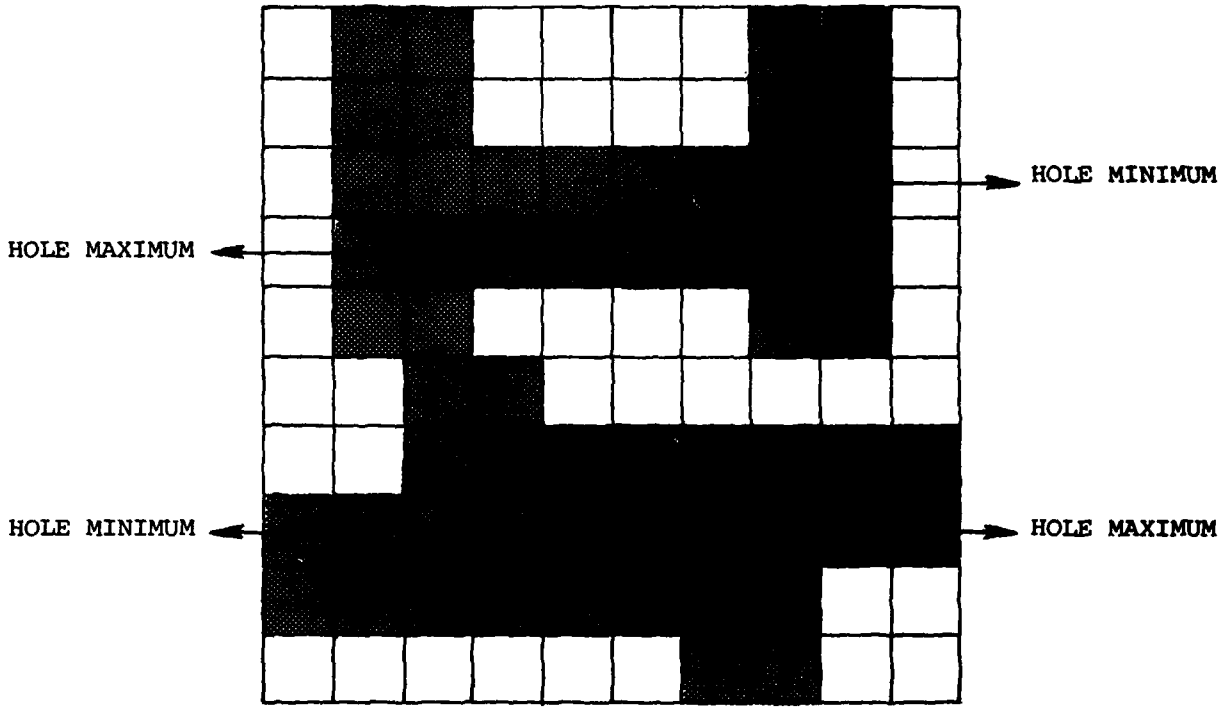


Figure 6-13. Junctions of Degree 3

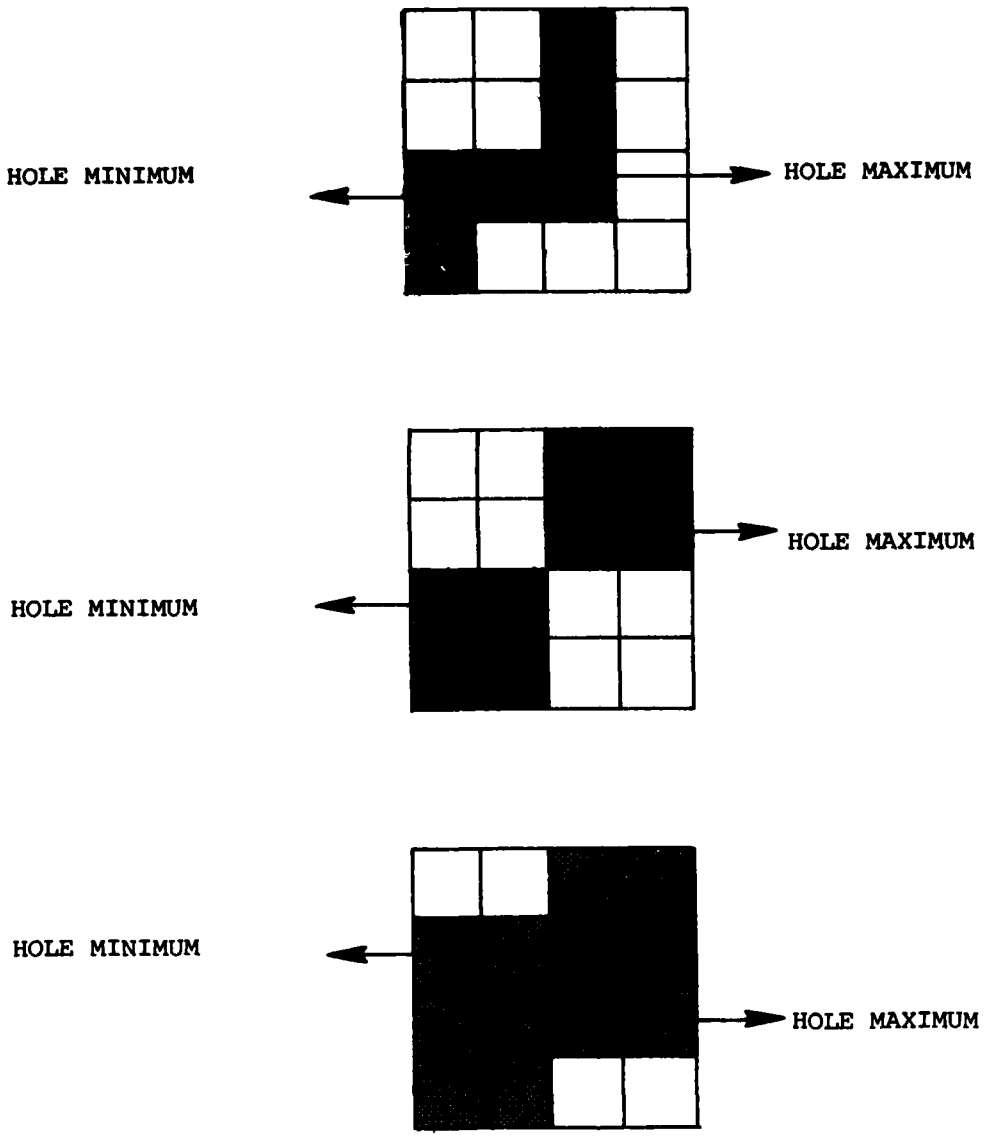
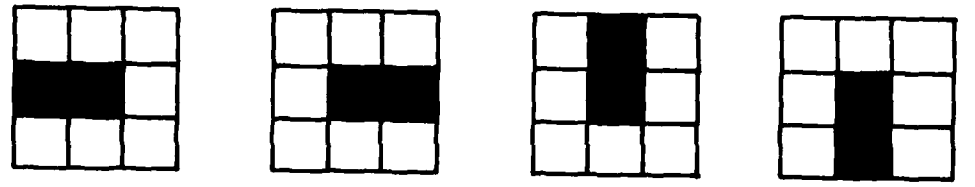
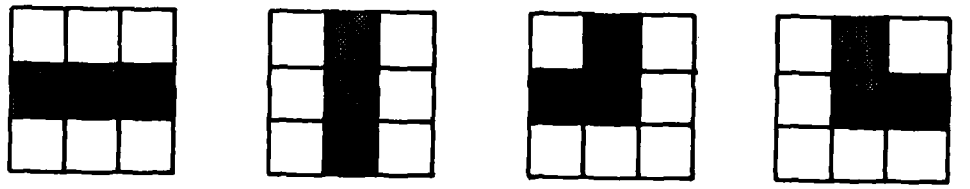
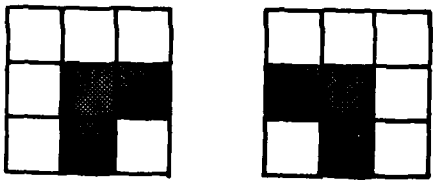


Figure 6-14. Non-Junction Ambiguities

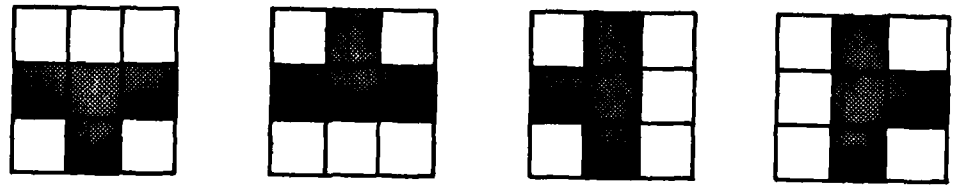


DIRECTLY CONNECTED PAIRS

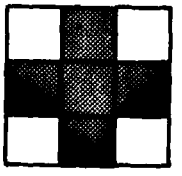




DIRECTLY CONNECTED TRIPLETS

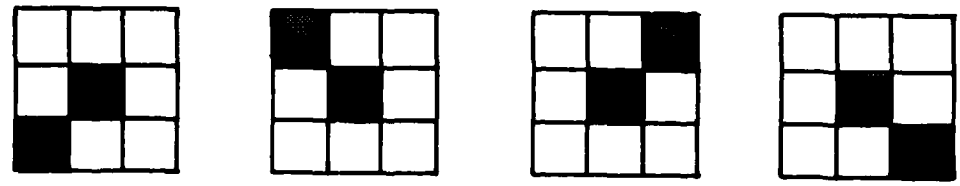


DIRECTLY CONNECTED QUADS

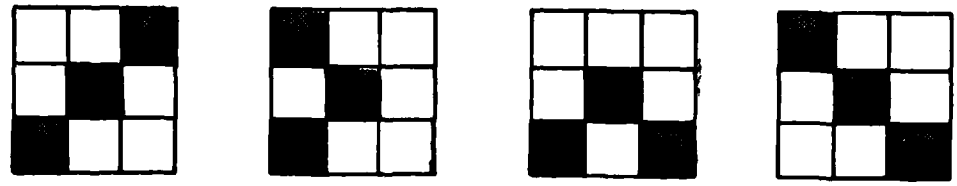


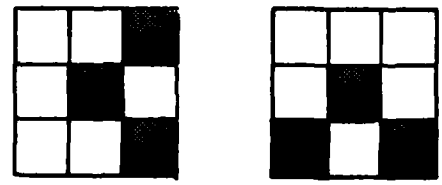
DIRECTLY CONNECTED QUINTUPLET

Figure 6-15. Directly Connected Combinations in a 3 x 3 Neighborhood

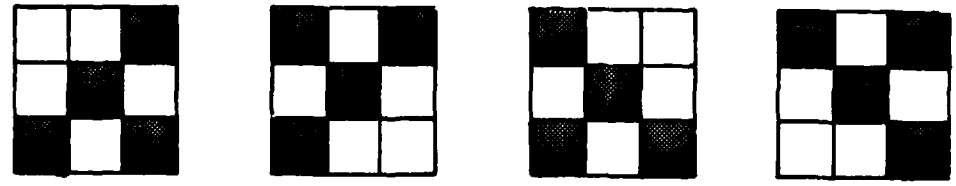


INDIRECTLY CONNECTED PAIRS

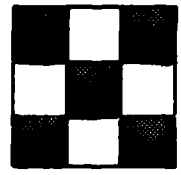




INDIRECTLY CONNECTED TRIPLETS



INDIRECTLY CONNECTED QUADS



INDIRECTLY CONNECTED QUINTUPLET

Figure 6-16. Indirectly Connected Combinations of a 3 X 3 Neighborhood

Examples of the three levels of resolution are shown in Figures 6-17, 6-18, and 6-19. Multiple levels of resolution offers flexibility in interacting with a raster image. In some cases a cartographic manipulation applies to a feature in the larger sense. In other situations finer subdivisions of feature segments are more appropriate.

An 8-bit word, of course, allows 256 labels which may be exceeded in some cases. Certainly a 16-bit word, with a maximum label value of 32,768, would be sufficient to retain all label numbers for a complex image at the finest level of resolution.

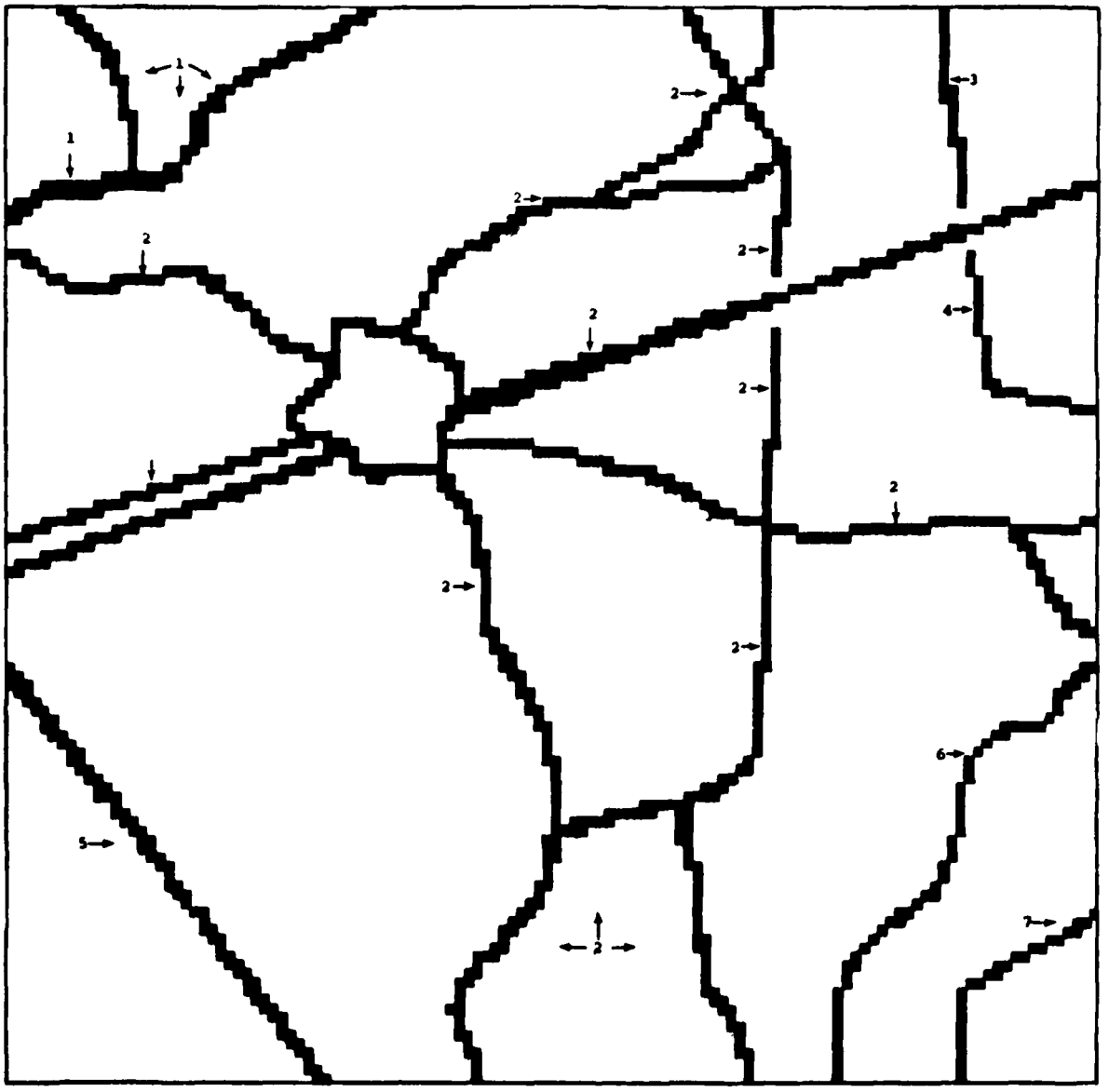


Figure 6-17. Directly and Indirectly Connected Labeled Components

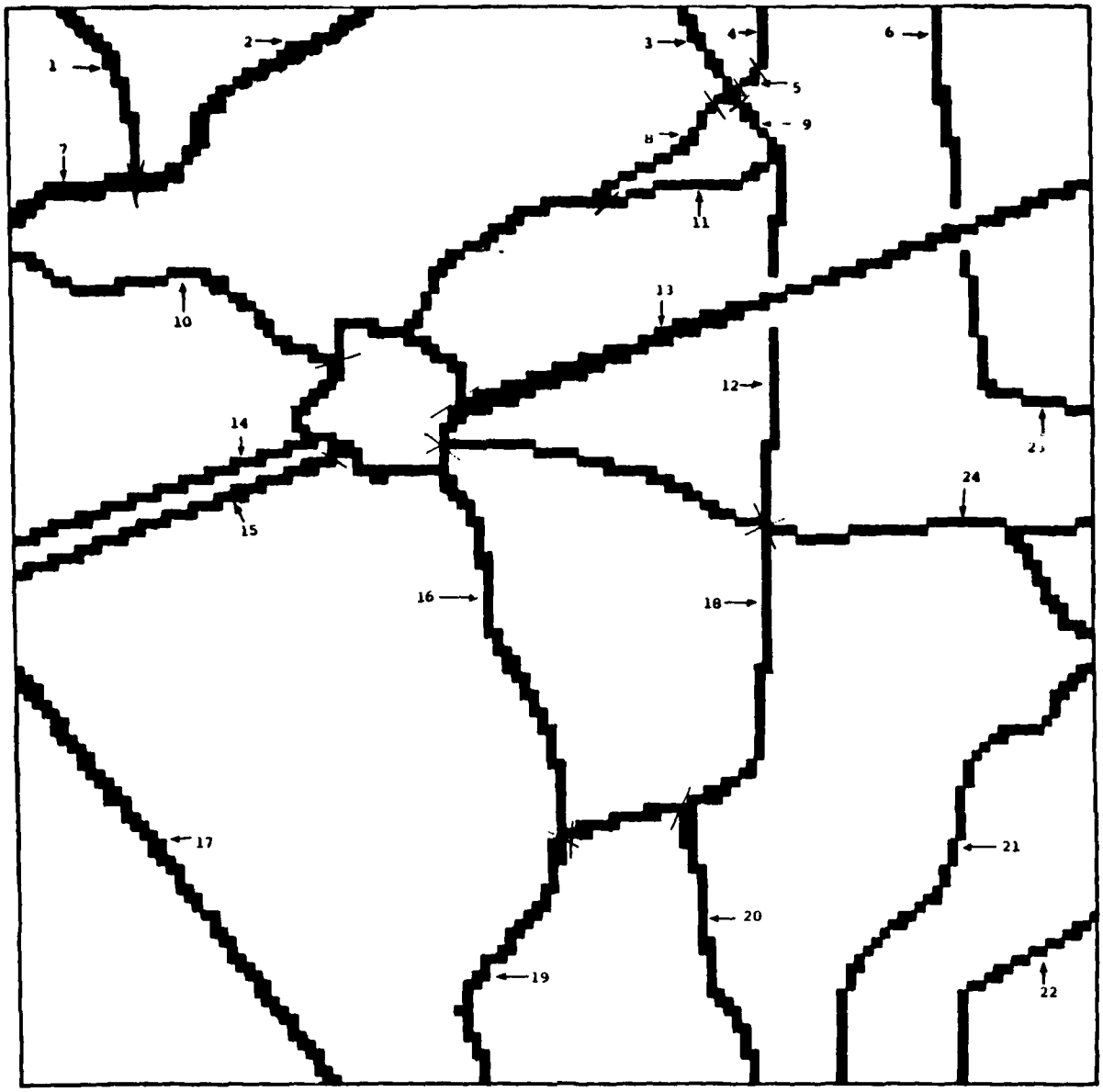


Figure 6-18. Directly and Indirectly Labeled Components Spatially Segmented at T and X Junctions

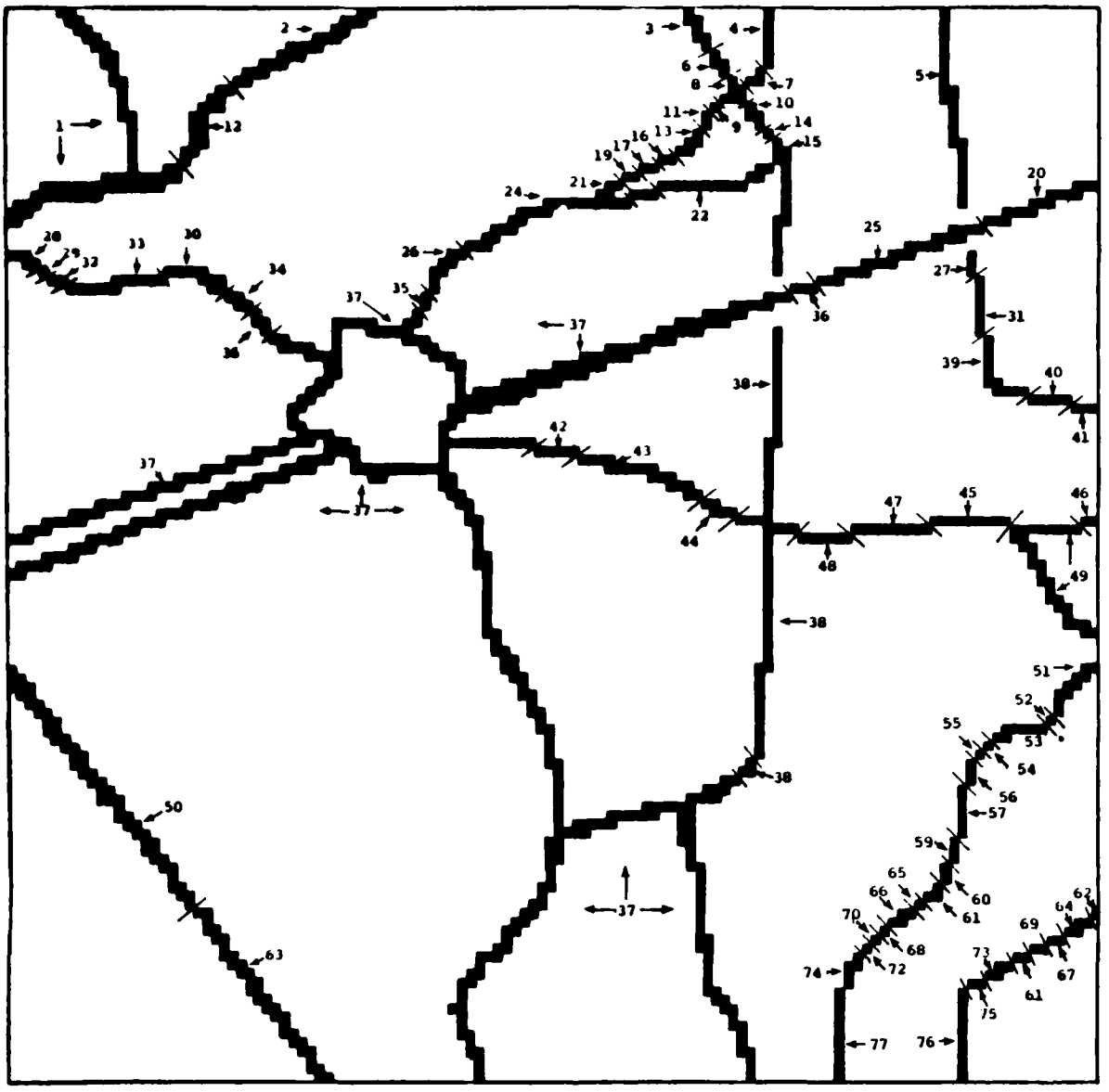


Figure 6-19. Directly Connected Labeled Components

SECTION 7. CONCEPT OF OPERATIONS

This section contains a description of the data flow, software processing, and software environment for cartographic manipulations of raster data. Figure 7-1 is an overview of a conceptual raster cartographic system within the chart compilation and revision environment, which is the main context for our data structure. Prior to cartographic manipulations, the processes of segmentation, PCM encoding, and labeling are performed. Segmentation forms subsets of pixels on the basis of color and halftone density and gives each pixel a property class number. PCM encoding examines the 3x3 neighborhood of a center pixel and encodes neighboring pixels with similar property classes by setting a positional bit in the Pixel Connectivity Map. Labeling forms connected components by partitioning a segmented pixel class and assigns each connected component a unique identifying label. Cartographic manipulation includes the processes which review, enhance, and correct the cartographic raster data. Merge/Conflict Resolution recreates the visual array and resolves conflicts among coalescing pixels.

With the exception of the segmentation necessary to form the initial color classes from the raster scanned source material, these processes are part of an iterative cycle. These operations would be performed each time an operator specified an item or feature to review or edit. A typical sequence might proceed as follows:

1. An operator indicates that he/she wishes to edit the water bodies in a selected area of interest. This specification would be input by typing the values of the various color codes which are being used to represent the shades of blue on the source material, such as color codes 10 and 12; pointing to the representative colors, either features or a menu, or by association of pseudo names, such as rivers and lakes, specified from a menu or by keyboard entry.

2. A PCM is assembled for every pixel in the area of interest for the class of interest based upon similarity of pixel values. Pixel values of 10 or 12 would be treated equal and synonymous.

3. Object pixels are further partitioned by assigning unique labels based upon one of the three levels of resolution. All background pixels are left unlabeled.

4. The desired cartographic edits are performed, adding a tributary, or moving the placement of another river.

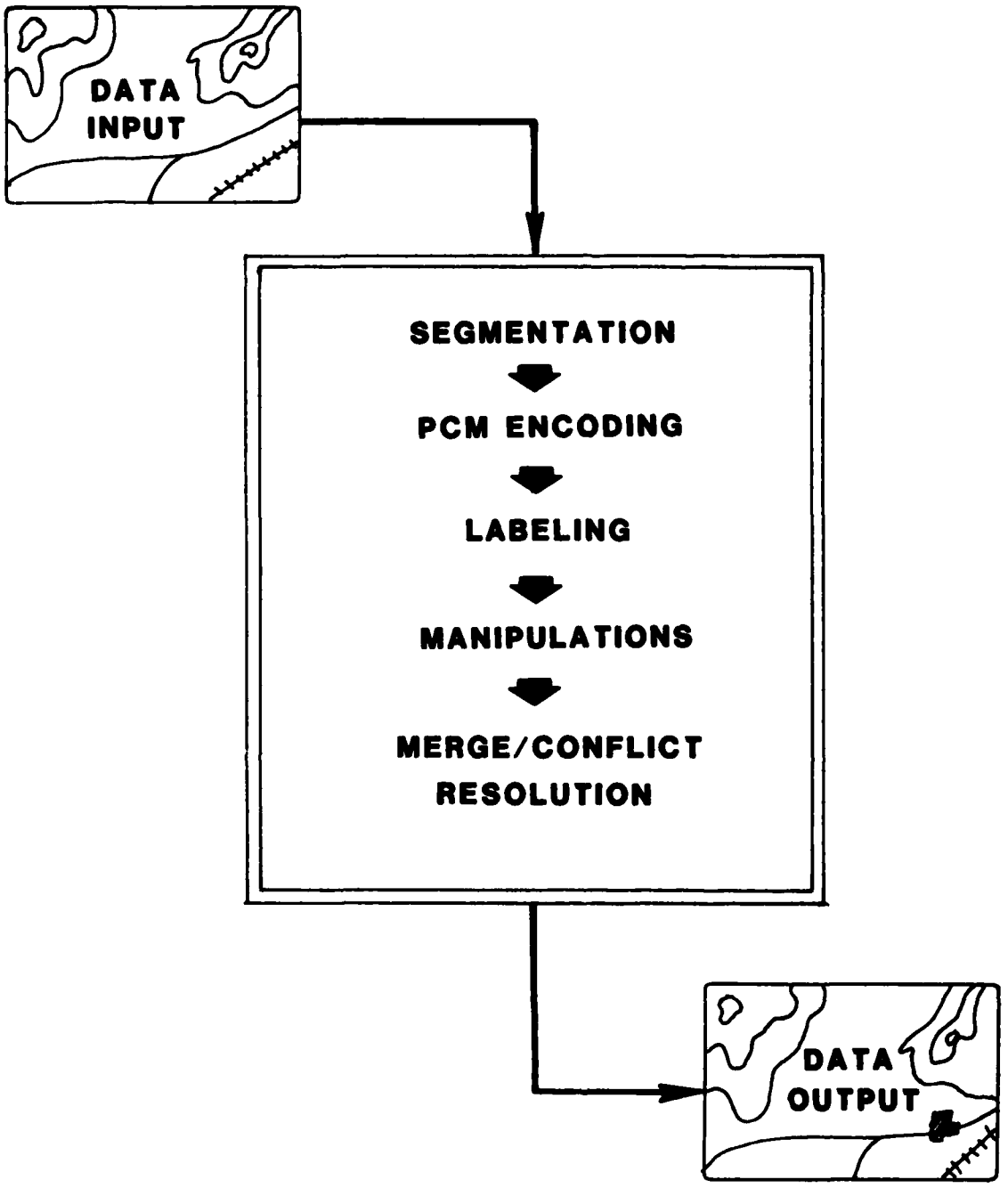


Figure 7-1. Concept of Operations for a Raster Cartographic Compilation System.

5. The pixels which have been overlaid with the tributary would reflect a new pixel value. The relocation of the river would also require repainting the pixels left unoccupied.

6. Additional edits are performed by indicating a new area and/or features of interest, in which case the process is iterated.

7.1 Segmentation

Segmentation of an image is the identification of subsets of an image where each subset is a contiguous region of pixels differing in characteristics from neighboring subsets. Segmentation of a raster scan is done by first preprocessing the raster scan to form a 'cleaner' image and then applying the segmentation techniques. The segmentation techniques can be generally classified as either intensity based, such as thresholding and clustering, or spatially based, such as edge detection or region extraction. An overview of segmentation techniques is given in Figure 7-2. Although spatial segmentation is important, the thrust of segmentation in our data flow is intensity based.

Thresholding takes an input continuous function and outputs a set of discrete quantized levels, (i.e., all input values within a range are assigned a single value). Thresholding may be done based on statistical or clustering techniques.

The statistical techniques may be:

1. Global - where the thresholds are assigned based on the entire image. This type of thresholding is used by the intelligent scanners such as SCI-TEX.

2. Neighborhood - where the local property of the pixel neighborhood such as average grey level is used.

3. Dynamic - where the location (x,y coordinate) of the pixel determine which global and neighborhood threshold function will be applied.

Structural thresholding has been done using two techniques:

1. Textural recognition where variances in textural pattern are recognized even though the intensity or spectral characteristics may be equivalent over a neighborhood.

2. Syntactic recognition where apriori knowledge is applied to recognize certain shapes within the image.

Clustering is the multidimensional extension of thresholding where two or more characteristics are used to group points in feature space. This is a technique which has been applied directly to the multispectral image. Many other combinations have also been used especially where textured and low gradient images are segmented.

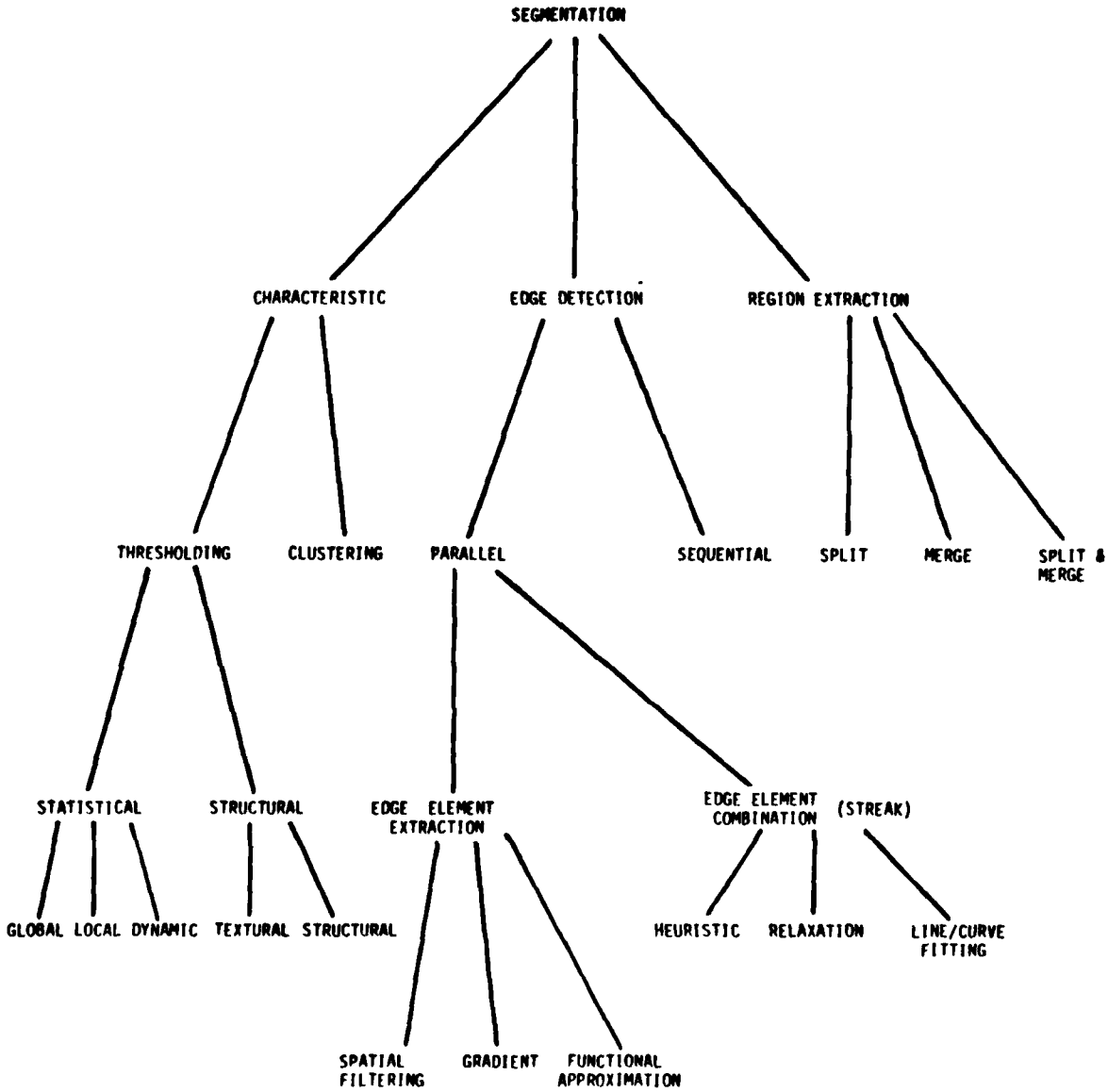


Figure 7-2. An Overview of Segmentation Techniques

Simple global thresholding allows specification of pixels of interest, as discussed in Section 6, and is suitable to parallel processing. Simple thresholding operators recommended for use are illustrated in Figure 7-3.

7.2 Encoding the Pixel Connectivity Map

The encoding or creation of the pixel connectivity map can be done by a software function on an ordinary serial computer, by using the function shown in Figure 7-4.

However, the simplicity of the operation allows use of a hardware encoder which can accomplish this task rapidly in one pass through an image (Sobel, 1978). By treating an NxN binary image as a bit stream with its origin in the upper left, a complete image can be stored in 2N+3 bits. Any 3x3 neighborhood can be accessed as subsequence using a shift register as shown in Figure 7-5. Full 3x3 neighborhoods can only be described for an (N-2) x (N-2) portion of the image array. Array border conditions force the encoding process to zero all border pixel PCMs for the following four conditions:

1. Beginning column of scan line.
2. Ending column of a scan line.
3. Initial scan line row.
4. Final scan line row.

A counter can keep track of the position of the moving 3x3 window and reset the shift register to zero the appropriate PCM bits. The use of the hardware encoding simplifies pixel accessing and extraction. By using high performance bipolar RAM for pixel storage and accessing, the computations necessary to encode PCMs for an image are realizable in video frame rates. (1/30 second).

7.3 Labeling

The purpose of connected component labeling is to further segment pixels of a similar class and to provide for an association of pixels belonging to the same entity. This labeling process creates feature segments which may or may not provide an exact correspondence to logical cartographic features. The remainder of this section will describe various labeling algorithms and recommend a specific labeling technique. Finally, the labeling process interprets the PCM to create multiple levels of resolution. The final section discusses methods for supporting this capability.

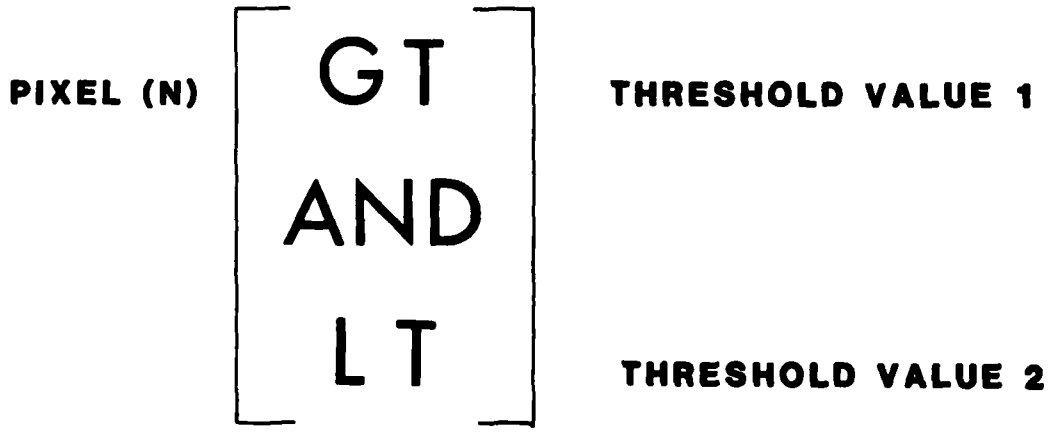


Figure 7-3. Thresholding Operators

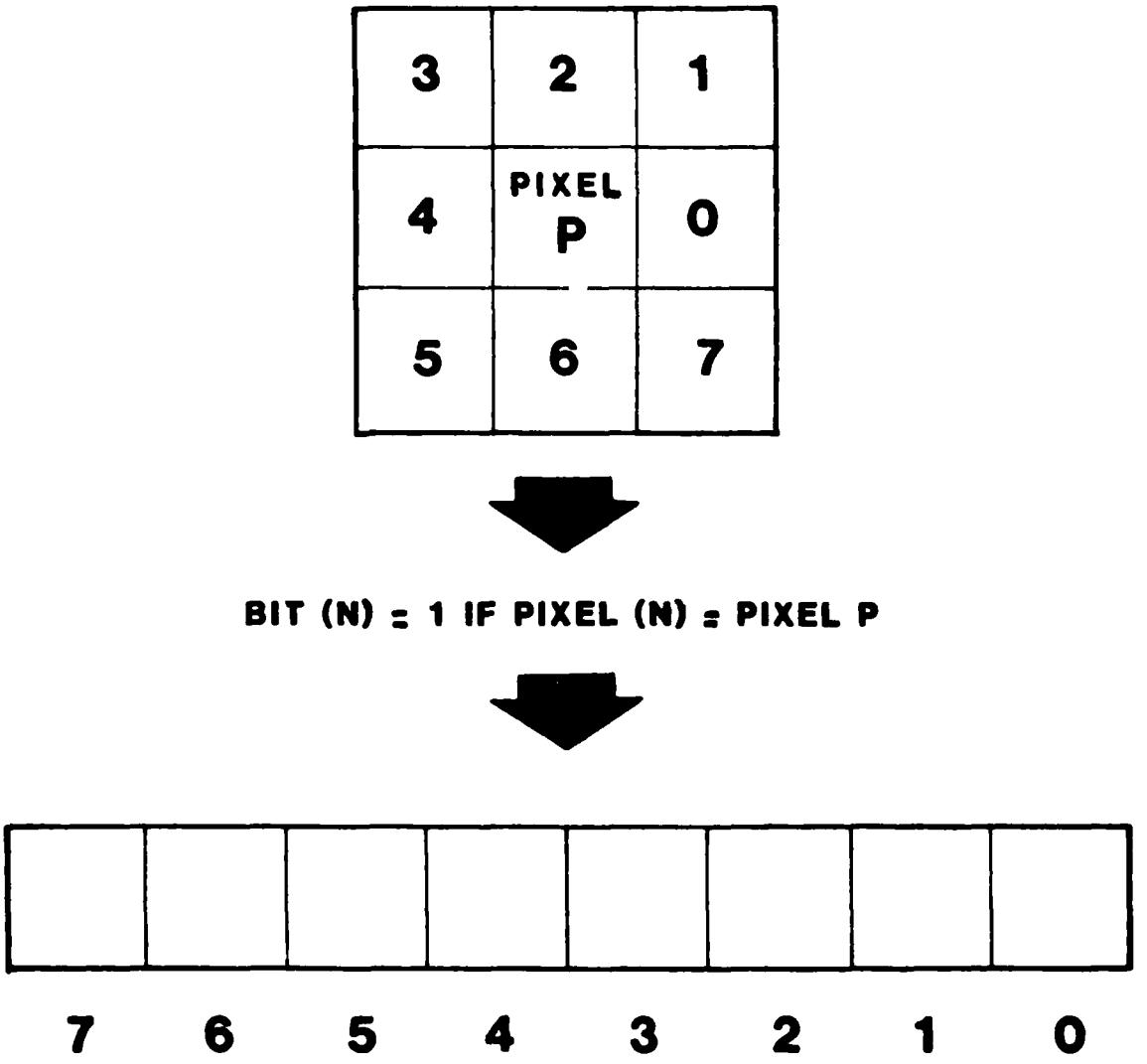
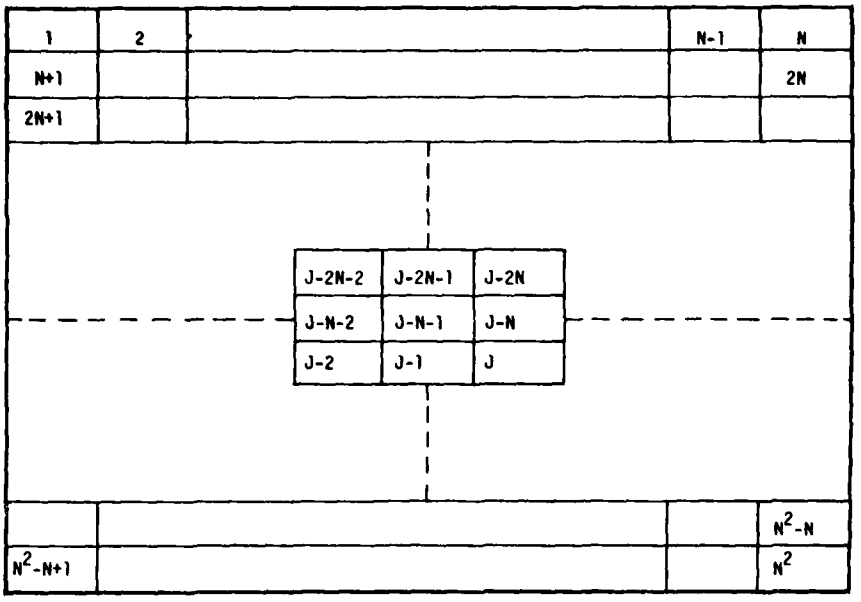
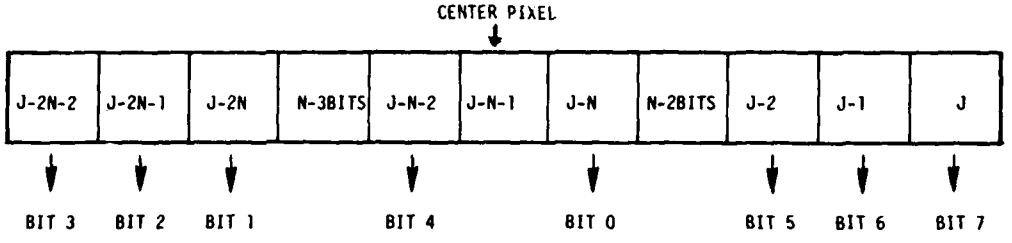


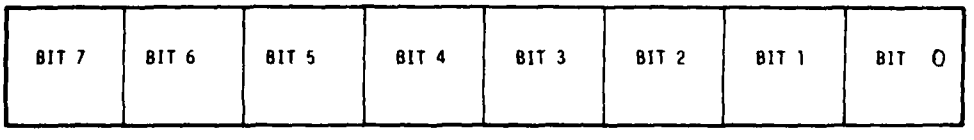
Figure 7-4 Encoding the Pixel Connectivity Map



NXN BINARY IMAGE WITH MOVING 3X3 NEIGHBORHOOD WINDOW



SHIFT REGISTER WITH CONTENTS OF 3X3 NEIGHBORHOOD



OUTPUT FROM THE SHIFT REGISTER IS MAPPED INTO PCM BIT POSITIONS

Figure 7-5. Using A Shift Register to Encode PCMs

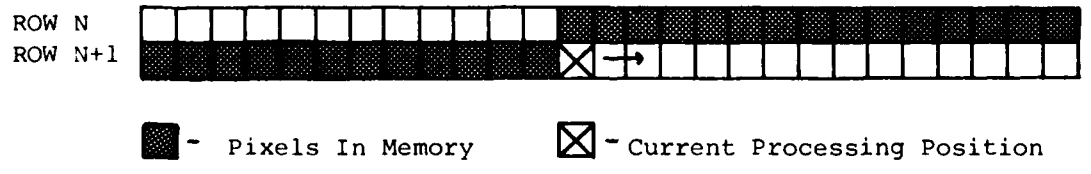
7.3.1 ROSENFELD'S SEQUENTIAL LABELING ALGORITHM

An early article by Rosenfeld and Pflatz (Rosenfeld, 1966) reported a sequential algorithm for labeling connected components. The top-down, left-to-right algorithm considers a single pixel (A_ij) in a binary array and outputs an array of uniquely labeled components. The labeling process is comprised of three cases as shown in Figure 7-6 . In case one, a background pixel remains unlabeled. In case two, an object pixel is given a unique label if neighbors 1, 2, 3, and 4 are background pixels. In case three, an object pixel has neighbors, 1, 2, 3, and 4 some of which are background or some are labeled pixels. In this situation the pixel is arbitrarily assigned to the smallest label. Differing label numbers are added to an equivalence table for later processing. Jarvis (1982) reports that relabeling can be done immediately but is more efficiently accomplished by marking the junction in an auxiliary array and relabeling from the equivalence table in a second pass.

One disadvantage to this algorithm is the labeling of pixels which may be unconnected. For example, an object pixel in case two could have neighbors 0-7 as background neighbors and be labeled. An approach to resolving this problem is given in (Rosenfeld, 1978) by a shrinking/ expanding operation which first changes all 1's to 0's if they have 0's as neighbors, then changes all 0's to 1's if they have less than two pixel neighbors (Figure 7-7 (Dyer, 1980)). Rather than eliminating isolated pixels, it is possible to interrogate the pixel connectivity map to examine the local neighborhood. An object pixel without connected neighbors can remain unlabeled, preserving the pixel, but eliminating it from considerations for further manipulations.

7.3.2 MILGRAM'S REGION TREE CONSTRUCTION

Milgram (1979) describes a labeling algorithm as part of a process to build a region tree and boundary chain code describing an image. The labeling algorithm scans with a 3x3 window comparing the current scan line to the previous scan line. Foreground objects are assumed to be directly connected and background objects are indirectly connected. As each point is processed it is compared to the labeled points in the scan above and to the left. Thus each point is assigned a previous label number or assigned a new number.



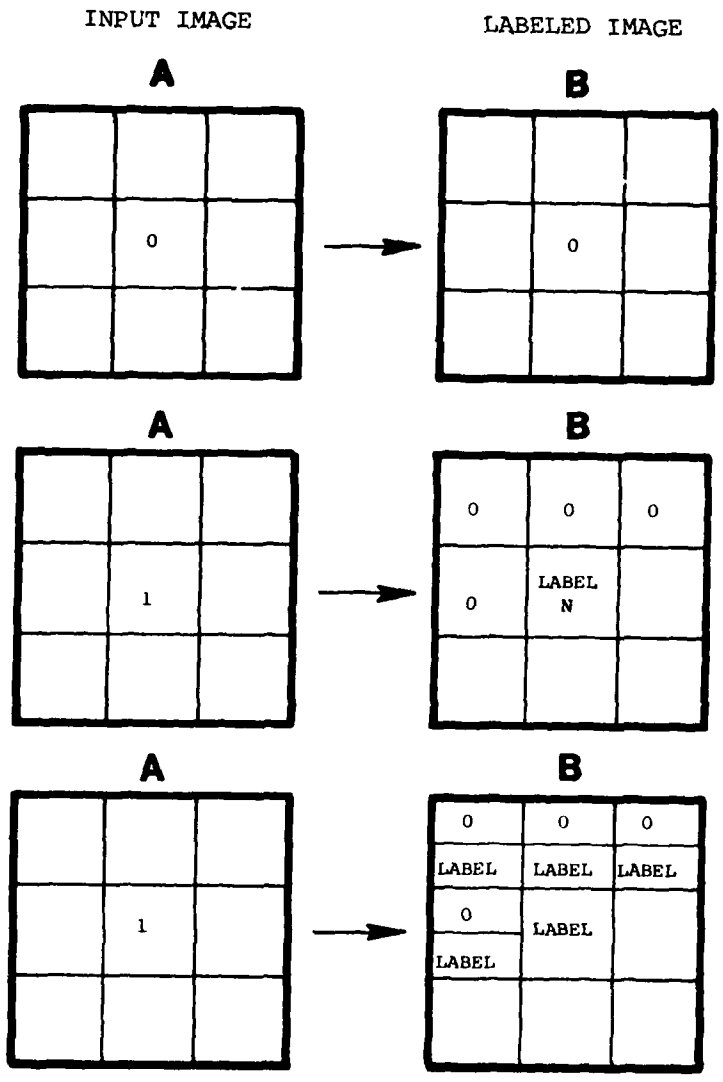
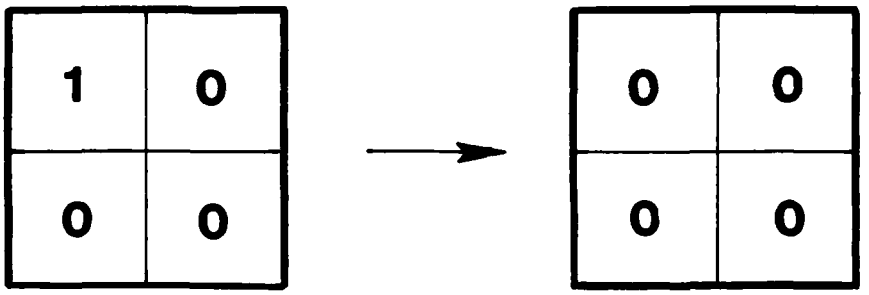
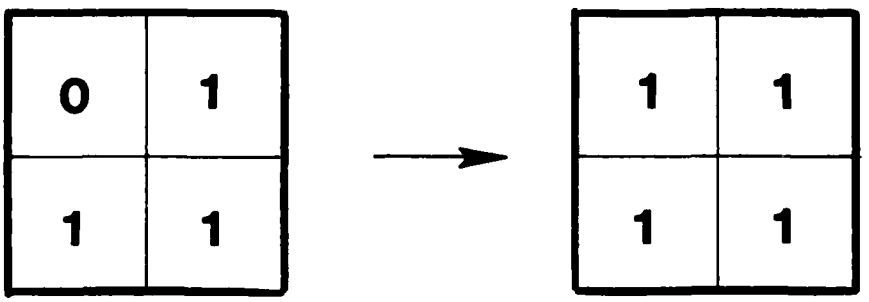


Figure 7-6. Rosenfeld's Sequential Labeling Technique



SHRINKING



EXPANDING

Figure 7-7. Noise Elimination

Active label numbers are retained in a list. At the end of each scan the newly assigned label numbers are matched to the active list. Those active numbers which do not match with the labels of the current line are considered to belong to terminated components. This information is used to decide region enclosures for constructing the tree data structure. It can also be used as a notice to terminate gathering statistics for a particular component. Finally, the terminated labels are dropped from the active list, the list is updated with newly assigned labels, and the current row becomes the previous row. Components which are later discovered to be subcomponents of the same region have their labels placed in an equivalence table to be combined on a second pass. Milgram's algorithm is recommended and presented as a listing in Appendix A.

7.3.3 CONNECTED COMPONENT LABELING USING QUAD TREES

Samet (1979) describes a two pass labeling algorithm which has three phases. The first phase explores and labels all possible adjacencies between a foreground pixel and its eastern and southern neighbors. In this pass only the foreground pixels are explored and due to the left-to-right, top-down nature of the scan, only eastern and southern pixels are examined. The process of examining adjacent neighbors entails traversing up the quad tree until a common father node is discovered and descending the tree to examine and label the adjacent pixel.

The second phase of the algorithm simply assembles all equivalent labels into equivalency classes. The third phase uses the equivalency classes to relabel pixels of the same component.

The running time of the algorithm depends upon the number of object pixels which must be examined by phase one, the shape of the components which causes equivalencies in phase two, and the number of equivalency pairs which need to be processed by phase three. Overall the average worst case time is proportionate to the product of the log of the image's diameter and the number of terminal nodes which describe the area.

One advantage to this algorithm is the lower number of equivalency classes which must be processed due to the phase one process of locating and labeling all possible southern and eastern neighbors. A disadvantage is the need and cost of constructing the quad tree prior to labeling. The tree structure can be a costly overhead if not used for other processes.

7.3.4 LABELING RESOLUTIONS

By applying multiple interpretations of connectivity to a PCM and/or searching the PCM's for component junctions it is possible to create labeled segments at several levels of resolution. These segment levels include:

1. Single labeled segments consisting of both directly and indirectly connected pixels,

2. Splitting the above segments at junctions of degree 4 and 3 (X and T junctions), and assigning two separate labels,

3. Single labeled segments consisting of only directly connected pixels.

The capability offered by multiple levels of segment resolution allows an operator to examine the area of interest at increasingly finer levels of resolution. For some operations this is an advantage. For instance, feature realignment may apply to two connecting roads, only one road, or some subsegment of a road.

7.4 Cartographic Manipulations

The manipulation of cartographic data has been the primary objective of our concept for an all raster cartographic system. Prior sections have discussed the desired functions, pre-manipulation processes and proposed data structure to achieve our objective. This section will attempt to explain how the desired functions would be implemented.

To attain an understanding of the procedures involved, the given data structure is reviewed to define the exact context in which the functions are presented. In addition to providing an effective data structure, there is also a need for mechanisms to support both human and machine decisions and the storage of information. Relative to our concept these mechanisms are unique files and buffers whose purpose and suggested organization will be thoroughly discussed. Finally, the remainder of this section deals with each function as described in section 5.1.1. What they are and how they are performed are discussed with respect to the given data structure and supporting files.

7.4.1 DATA ORGANIZATION

Upon completion of the pre-manipulation tasks of labeling and segmentation, the derived data would be loaded into three distinct memory planes. Each plane would consist of a specified area of RAM organized into a 1024 x 1024 array of eight bit elements. Each element in a single array would hold information about a corresponding pixel on a raster scan display. The use of RAM planes would permit the data elements to be accessed at video frame rates. This is desirable since a complete image is described within the memory planes which are directly and indirectly used to drive the video electronics. A common rate at which pixels are refreshed is 30 times per second. Instituting data changes at a corresponding rate would result in a consistent visual display of the exact contents of the memory planes. These faster, and in some functions, immediate responses will benefit the user and ultimately system productivity.

One of the first memory planes to be filled is the pixel value map. This is an array which will store the specific value of each pixel on the raster display. The screen coordinates of a pixel would serve as the address into the pixel value map. With each array element eight bits in length, two-hundred and fifty-six displayable colors are possible. Most raster displays are capable of producing many more colors. However, 256 is an ample number given that raster scanning devices are usually designed to distinguish 12 colors.

The pixel value map could be used for several different purposes. Foremost are the select and display functions where the map could be used to locate and extract desired data. To accomplish this, the map need only be scanned once. A background color would be sent to the video electronics at

each pixel location not having the desired color. This would result in the elimination of all but the desired color from the screen in one refresh cycle. The data would remain intact while the display is changed and could be restored in the same manner.

Other uses include statistical calculations such as determining the percentage of each color displayed. Conflict resolution and edit prompting are also functions which could be executed via the pixel value map. Examples of these will be given throughout the presentation of the cartographic function.

In another designated area of memory, the connected component label numbers will be loaded. Here also, the label numbers are stored as an array with each element corresponding to a single pixel. The label number would be stored as integer values in eight bits allowing for 256 unique label numbers. Initially this is foreseen as an adequate amount as the concept is designed for working on sections as opposed to complete cartographic manuscripts. If this factor is insufficient, the array could be expanded to 16 bits offering up to 32K unique label numbers.

By accessing this area of memory we can determine the finite set of pixels associated with any given pixel. Thus manipulations on groups of pixels can easily be performed through identification of a single pixel on the screen or knowing the exact connected component label. For example, moving a feature to a new location requires only one pixel to be identified along with the new position. From this the system can scan the array of connected components translating each pixel with the proper label to the new position.

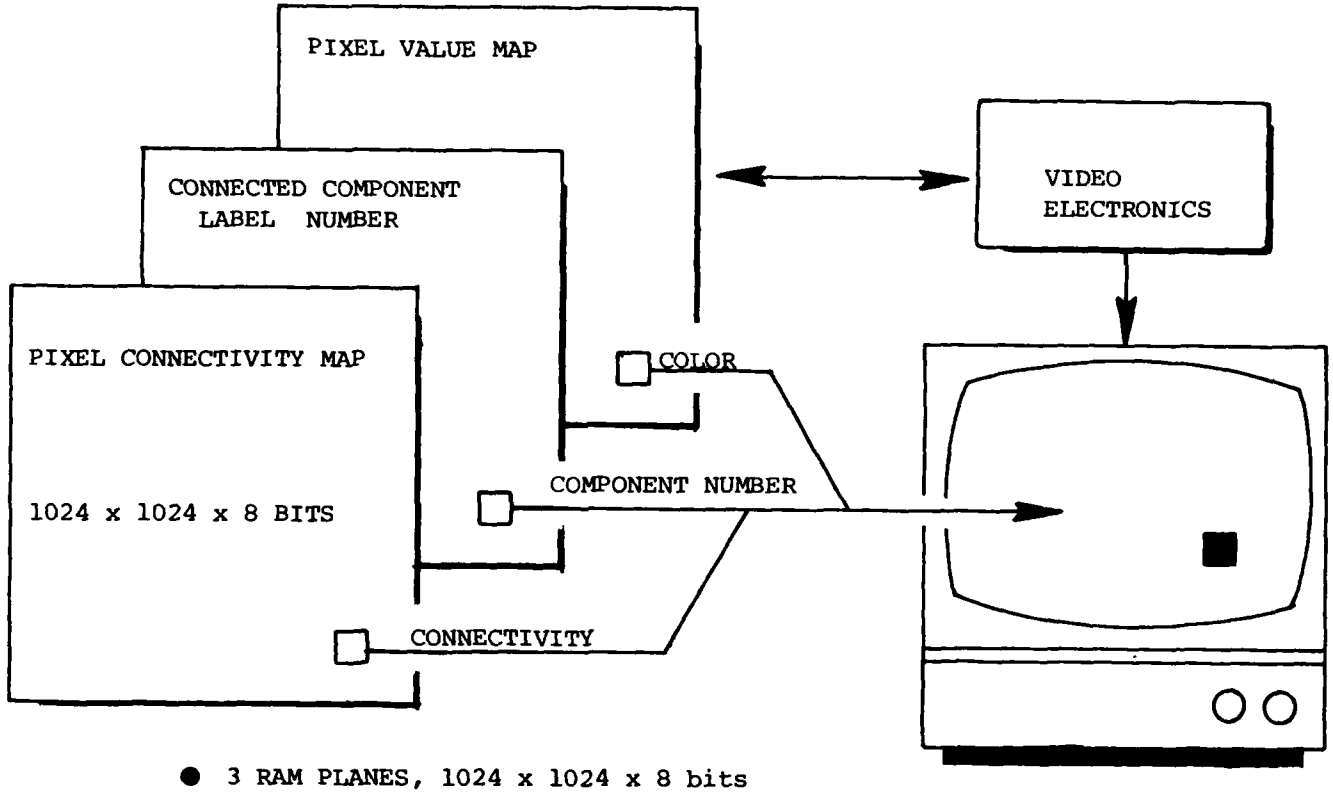
The third and final memory plane as defined by our concept is used to store the pixel connectivity map. Extensively discussed in Section 6, this area in memory is used to construct and store the relationship of surrounding pixels for each picture element. From this pixel connectivity map it is possible to determine edges, boundaries, and other characteristics which will facilitate editing and revision of the data.

In summary, the data is stored in three separate planes of random access memory. As shown in Figure 7-8 , for every picture element on the screen there is a pixel value, connected component label, and pixel connectivity map. Using the screen coordinate of a pixel as an address into the memory planes we will be able to ascertain the given pixels:

1. Color,

2. If it belongs to a collection of pixels representing a feature, and

3. The relationship of neighboring pixels.



- 3 RAM PLANES, 1024 x 1024 x 8 bits
- EACH LOCATION CORRESPONDS TO A SINGLE PIXEL ON THE DISPLAY

Figure 7-8. Data Organization

These data elements hold the unique information concerning a pixel, and provide the potential for successful cartographic manipulation in a raster format.

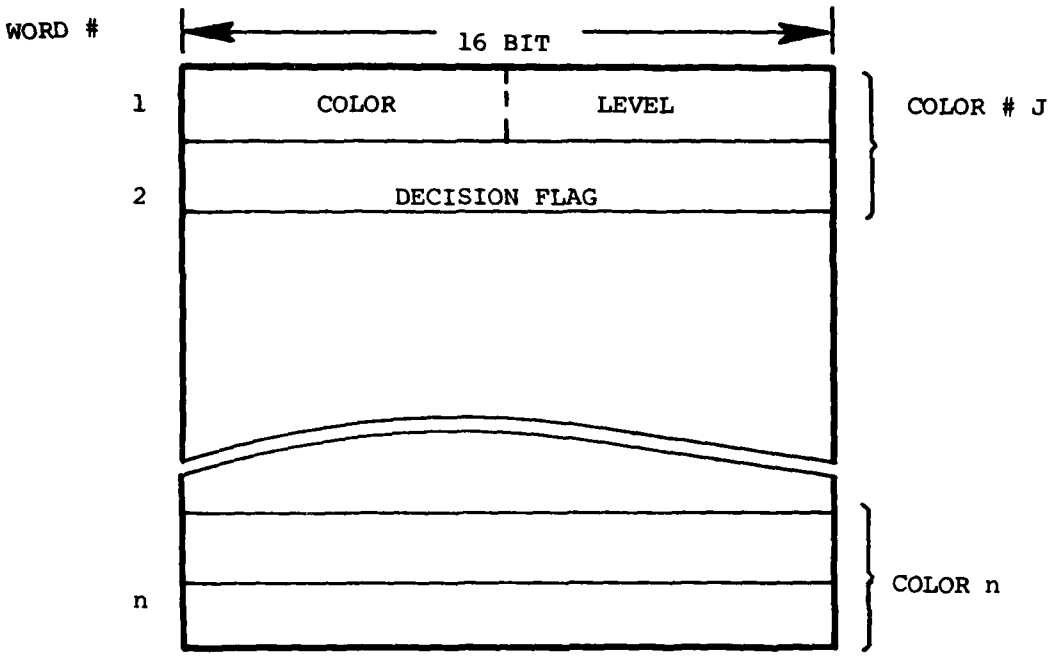
7.4.2 PROCESSING REQUIREMENTS

While our data structure supplies the framework to design functions it lacks the peripheral elements to make it functional. The data structure at this point allows no way of restoring data when the results of a manipulation are rejected. There is no provision for adding and changing textual remarks or drawing upon pre-built symbols instead of continually reconstructing them. Adding system expertise, even at a minimal level, requires more than just the given data structure. To overcome such situations and further enhance the advantages of our data structure, additional working files and buffers must be defined.

One such file or buffer is necessary to store all of the displayable colors. For each color there could also be a numerical value indicating its position in the hierarchy of colors. In this way the system could automatically perform simple conflict resolution tasks. Consider the case where a river is being inserted, or drawn over, the current data. If a road was crossed by the river, the color of river would be shown bisecting the road. Using a hierarchy of colors the system could easily detect what color is being written over. If black denoted the road and had a higher priority than blue for the river, then the system would not allow the river to be drawn at that location. Otherwise the river would be forced to update the road after the initial revision was made.

A suggested file or buffer organization is depicted in Figure 7-9. Since our concept allows for only 256 colors, the color value and its level in the hierarchy can be stored in a 16 bit word. Perhaps this could even be shortened by using the color value itself to indicate the level in the hierarchy. Figure 7-9 also shows a word reserved for a decision flag. This information could tell the system how to interpret the level in the hierarchy for a particular color. For example, a flag set to zero could indicate that the level should be obeyed always. A value of one might infer that the user wishes to handle all conflict resolution problems. With a decision flag stored as a 16 bit word, the user might also be able to store up to two exceptions. For instance blue might have a lower priority than black and yellow. However, the user may wish to manually resolve conflict resolution involving these three colors.

This hierarchial color file would possibly be created as a result of scanning the analog source. All the detected colors would be initially placed in the file. Upon entry into a manipulation session, an executive process would check to ensure that this file existed and was filled in. If not the user would be prompted to establish the hierarchical values.



WHERE: Color represents x value from Ø to 256. These values are also used in defining the pixel value map.

LEVEL represents a value from Ø to 256 defining the colors specific place in the hierarchy of color. All colors having a higher level would take precedence.

Decision Flags:

Ø = let user decide when conflicts arise

J = always

> J = exception

Figure 7-9. Color File Layout

To manipulate data and then restore the values if desired, there needs to be a way of recording the changes which have occurred. This could be accomplished by creating a temporary edit file for each manipulation that would not destroy the data in memory. This file or memory plane could store a component label and pixel value followed by a string of pixel addresses that are affected. Every time the data at any pixel was physically changed, the old values would be saved. The edit file would therefore store sufficient information to reconstruct the data as it existed prior to manipulation.

The edit file could even be expanded into an event file where all changes could be recorded. This would effectively permit users to step forward and backwards to review the changes made and the impact. Another possibility is to allow users to create edit files or event files as needed. This would speed up processing times as inappropriate data would not be saved. The essential point, however, is the need for a mechanism to allow users to reject the results of a manipulation. The use of a temporary edit file provides a simple solution.

A third file which may or may not be needed depending on the manipulations exercised is a textual file. If users want to build and modify headers and manipulate data based on header information, then a textual file is necessary. Such a file might contain a specific header number, some set of attributes, and the component label number.

This file may also permit additional expertise to be installed in the system. Consider the steps required to move a road. Identifying the feature by pixel address may be sufficient for the move function. However, examination of the header data may find that the selected pixel belongs to a feature represented as a dual cased road. In such a case, there may be additional moves required for pixels which are physically unconnected to the selected pixel. The system could detect this and notify the user or automatically make the move when the additional pixels are known.

The final file to be discussed is a symbol file. Cartographic data is represented by a defined symbol. Railroads, cities, rivers have standard colors and shapes which enable humans to readily grasp meaningful information. Performing edits/revisions will require users to create such entities. To continually place a feature and construct its symbolized representation is inefficient. A more effective approach is to identify the feature being created and its location on the display. The system should have the capability to construct the symbolization for the user. To allow for this, a symbol file is required in which the user can define and build symbols to be called up for later use. Every time a new symbolized feature is inserted, the system could access this file, retrieve the graphic display instructions and construct the visual representation of the feature.

In brief, our data structure is capable of storing all the required information for manipulation purposes. However, we need other mechanisms which will allow us to exploit the data. Some of these mechanisms include the files and buffers suggested here. Although they may not provide a complete description they offer a point from which to begin. The actual implementation would determine more specifically the files and structures needed. Figure 7-10 depicts these suggested files in relation to the data. Combined with our concept for a data structure these few files will provide for effective cartographic manipulations.

7.4.3 MANIPULATIONS PRESERVING DATA UNIQUENESS

In Section 5.1.1 we identified several manipulation functions which would not physically destroy the data. Restated here, we define these functions as the processes which will not destroy the contents of the RAM planes storing the data. This includes the procedures for selecting and displaying features as well as utility functions. In essence these functions view and interpret the data to output a result. Our visual perception may be altered or some new information may be generated but the data remains intact.

7.4.3.1 Select Functions

The first step in performing an edit to existing data is to identify the entity to be modified. A selection function can provide this capability but must accommodate individual features as well as entire classes. Utilizing component labels or pixel values, four options are possible:

1. Individual features selected via the cursor position,

2. Individual features selected through a specified component label,

3. A class of features selected on the basis of color,

4. A class of features selected by a range of connected component labels.

The pixel connectivity map would not be generated at this function level nor as a result of the subsequent displaying of selected features. The reason for this is that a user may select features repeatedly without manipulating them. To build a pixel connectivity map under such conditions represents unnecessary processing.

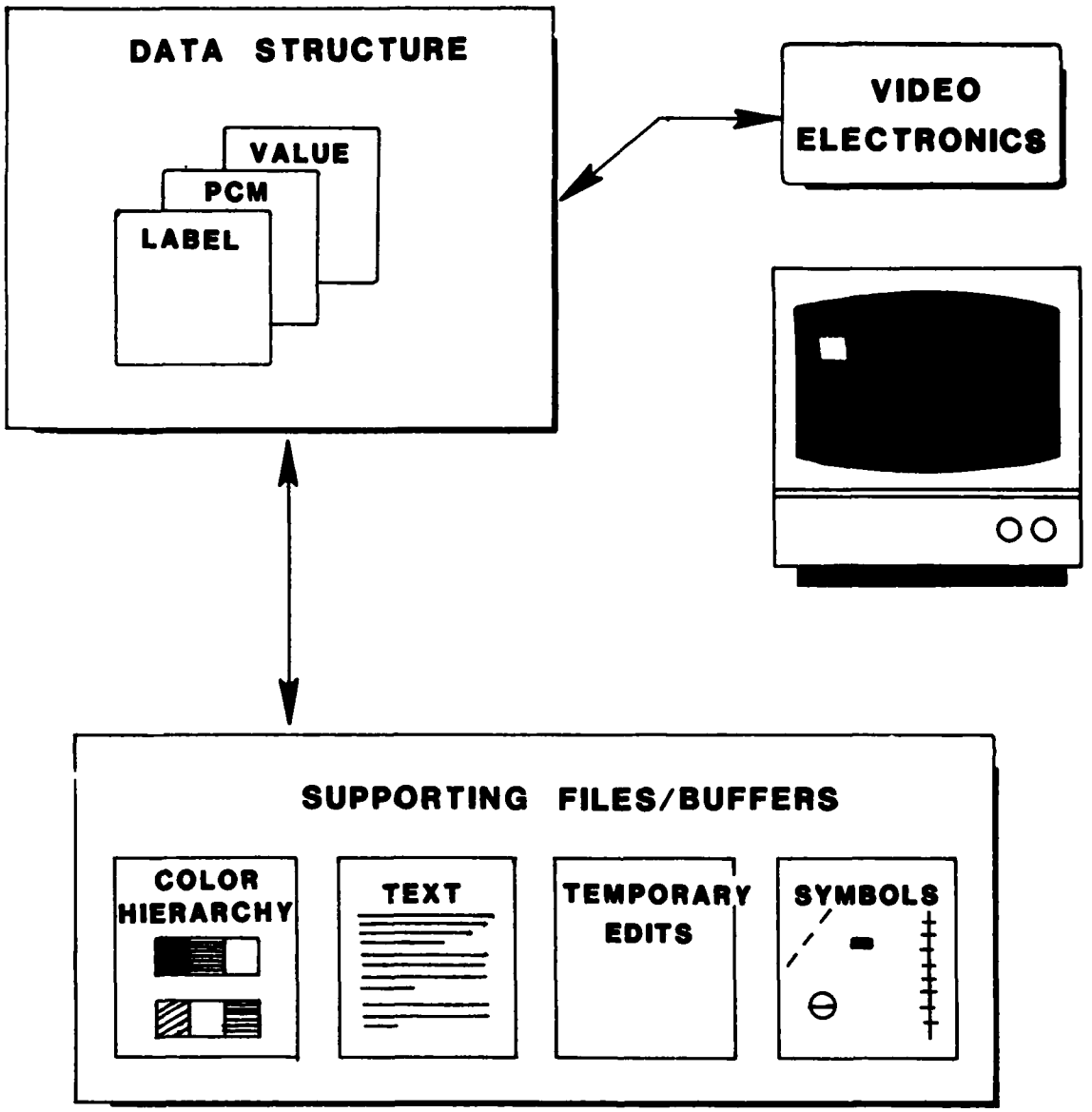


Figure 7-10. Elements for Effective Raster Cartographic Manipulations

Selecting a feature via the cursor requires several steps. First the cursor must be enabled followed by the users positioning of the cursor on the desired feature. In cases where the display is crowded the user might utilize a zoom function to aid in selecting the appropriate pixel. Another aid in resolving such a problem is discussed under the display functions. Once the cursor has been positioned, the user would execute a select by coordinate position command. This and other functions might use a function keyboard or menu driven displays to receive user directives. Regardless of the method used to issue commands, once received the system would read the cursor position on the display. The corresponding element in the array of connected component label numbers would be accessed. The value stored there would be read and passed on to the display function (to be discussed in the following section). This component label number which would identify a single entity could further be manipulated.

From this description it is obvious, that if the label number was known then the cursor position is irrelevant. Users could simply execute a select feature by label number task. The system would prompt the user for a value and check its validity against a known range of values. The array of connected component labels would be scanned until the given label was found. In turn all locations with the matching label number would be directed to the display functions.

An entire class of features could be selected on the basis of pixel values or color. The user may position a cursor on a desired color and issue the command to retrieve all entities with a matching color. The system would respond by using the cursors position to access an element in the pixel value map. This value would be read and the entire pixel value map scanned. Each pixel having the same value as the selected pixel would have its location transferred to the display functions.

The fourth possibility is to select a class or group of features specified as a range of component labels. To execute this operation, the user would issue the appropriate directive. The system would query the user for a range or possibly a set of label numbers. Upon entry of each label the system should validate the given numbers. Again, the connected component labels in memory would be scanned. Location of labels meeting the condition of the range of values would be sent to the display routines.

A final possibility unrelated to the RAM memory plane is the selection of features by headers. If a header file exists then a user might only be required to enter header attributes. Comparing these attributes with those stored in the textual file would result in the selection of specific entities. Within the header data, a component label number would be read. This value would be scanned for in the same manner described earlier.

7.4.3.2 Display Functions

Display functions will permit users to display features, provide visual enhancements and perform display transformations. Likewise, all manipulations affecting the visual presentation of data will have to interface with these functions. The primary data that will be used in the display processes are pixel locations and color values obtained from the pixel value map.

Probably the first thing to be displayed is the data derived from the scanning process. To display all of the data a command such as "display source" might be given. The system would then clear the raster display and begin accessing the pixel value map. The value at each array element would be read and sent to the display output processes. The value would be interpreted most likely through a video lookup table and the color generated at the correct pixel location.

To display individual features as classes the user could be provided with two options. Features could be displayed individually by first clearing the display to a background color. As each pixel location in the feature is reviewed by the display routines, the corresponding color from the pixel value map would be retrieved. The pixel locations and color values would be transferred to the video electronics to produce a single feature on the display.

In some applications the user may desire to view features as they exist on the source display. The system could first paint the display as if a "display source" command was given. Then the desired feature or class would be highlighted. Higher display intensities, contrasting colors, or blinking would effectively offset the desired features from the source data. Such techniques provide visual enhancements and can greatly increase the effectiveness of the display functions.

Display transformations such as translation, rotation, scaling, and zooming are part of the display functions. Some devices offer these and more functions as hardware options providing extremely fast execution times. An advantage of transformations performed in hardware is the retention of data in its original form. Software transformations would require an intermediate area, like the edit file, to hold the transformed values for display purposes.

Other display functions not acting upon the data structure would include modifications to video lookup tables and intensity levels. These functions affect the way in which the raster output is viewed and modified through software control.

7.4.3.3 Utility Functions

Throughout a manipulation session a user may be required to ascertain information which is not readily available. Lineal measurements, area determinations, orientation angles and statistical calculations are a few examples. Our concept for a data structure would permit such functions to be implemented thus incorporating additional capabilities into the all raster system environment.

If actual map scales are known, then these values could be applied to the displayed data. A user might determine a measurement in the given map scale by identifying two cursor positions. The line defined by these two points could then be transformed to a map measurement. Similarly a user could enclose an area on the display by using the cursor in a tracking mode. With each pixel representing an approximate area on the surface of the earth, the square footage could be calculated. This would enable users to obtain information such as the size of cities, lakes, and reserviors.

Another measurement that could be taken is the percentage of each pixel value displayed. Since there is a known number of pixels (e.g.1024 x 1024 a ratio could be established between the number of pixels of a certain value and the total. A single scan through the pixel value map would retrieve the necessary information. A possible use is the amount of surface area covered by roads or lines of communications.

The angles at which symbols or features are displayed may also provide information. A user may wish to rotate some entity a given number of degrees from the current position. Perhaps the user could use the displayed orientation angle to determine if the entity correctly placed on the manuscript. For example, to determine the orientation angle of a given character a pixel connectivity map would be generated. With connectivity established for a specific character the exterior pixels can be located. The exterior pixels could be mathematically graphed and the angles of the sides of the character calculated algebraically

At some point the user may wish to know the number of features or classes. The highest value in the connected component label array would indicate the unique number of components. Likewise the highest color level in the hierarchy of colors would define the number of different classes.

Utility functions offer assistance to the user in many ways. Throughout the life of a raster cartographic system, different utility functions could be designed and implemented as needs change. They may be as simple as

allowing users to interactively modify video lookup tables or be so complex as to require execution in a batch mode. The list of utility functions could be expanded for clarity. We provide a restricted explanation of only a few examples.

7.4.4 MANIPULATIONS INVOKING VISUAL MODIFICATIONS

In the previous subsection we discussed manipulations which do not destroy data integrity. Contrasting these manipulations are those functions which alter data values. Cartographic functions which produce a physical change to the data can be separated into two categories. The group discussed in this subsection relates to functions which produce changes affecting visual characteristics.

7.4.4.1 Delete

The deletion of a feature requires the removal of all the information describing the entity from the data structure. Thus the pixel value, the pixel connectivity map, the connected component label, and any textual data associated with the feature will be affected. To perform this operation the feature in question must be identified through one of the select functions. Selecting the feature will produce a connected component label number. The display functions would then highlight or output a new display of all the pixels having the same component label.

To perform the delete operation the following procedures could be exercised. The first response by the system to the delete command would be to generate a pixel connectivity map using the pixel value (color) of the feature. A temporary edit file would also be generated at this time. The connected component label number and the pixel value of the selected feature would be placed in the temporary edit file.

The next sequence of events would process affected memory planes and visually remove the feature from the current display. The process begins by scanning the memory containing component label numbers. When the label number is detecting its location is saved in the edit file and the edit file word pointer incremented by one. The label would then be replaced by a zero indicating the absence of a unique component. The same memory location would be used to access the corresponding element of the pixel value map. Here a null color denoted by some value other than the current list of color values would be written. The same address would then be transferred along with the null color value to the display routines. The display tasks would then move to the given pixel address and generate the null color value eliminating the previous color. This process would continue until all pixel locations having the identified component label have been removed from the data structure and visual output. Once deleted, the user would be

queried for an accept or reject command. If the user was unsatisfied with the results then a rejection could be issued. This action would cause the system to reposition the pointer in the edit file and begin reading data and inserting it in the data structure and the raster display device. In effect, the deleted feature would be restored exactly as it appeared prior to the deletion. If the deletion was accepted then the temporary edit would be deleted and the user prompted for the next command.

If the procedures just described were followed then the user would have to go back and fill in the null areas on the display left by the deleted feature. This may be the only desirable method of performing deletions when a feature coincides with several different classes of features. However, if the deleted feature was entirely contained by another feature then a deletion by color change is possible. An example of this is shown in Figure 7-11 where the entity to be deleted is a river and is contained by an area of vegetation. The same deletion procedures described earlier would be followed except:

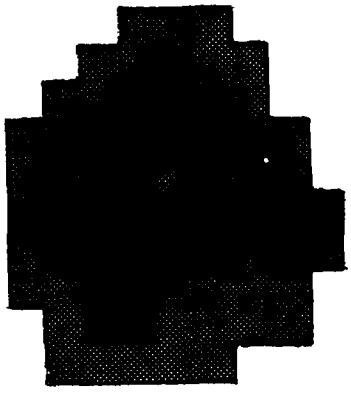
1. The pixel value for the vegetation would be used to replace the pixel value for the river as opposed to a null color value.

2. Instead of a null component label, the label number of the vegetation would be used.

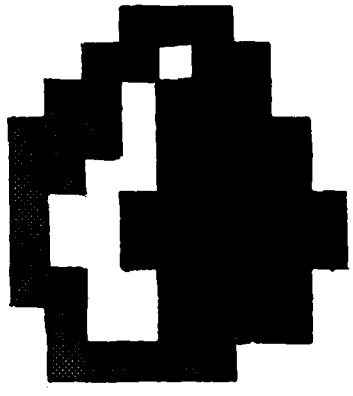
This technique could effectively delete the desired feature and automatically update the data structure and display. The user would be relieved from going back and filling the void areas created by the deleted feature.

A level of expertise could be added to this function by permitting the system to determine if a deletion by color change is permissible. Using the generated pixel connectivity map, pixels adjacent to the given feature could be tested for color. When all adjacent pixels are of the same color then the current deletion process is a candidate for the color change technique. The user would also retain the power to override the systems decision and manually fill the void areas.

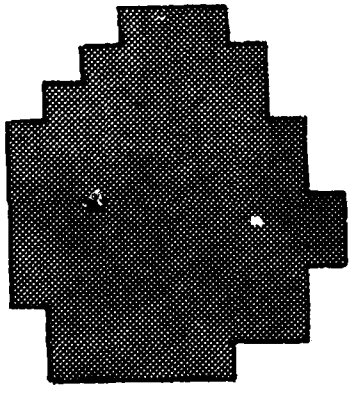
Features might also be deleted on the basis of header information. User might opt to define a header or a set of headers which the system could search for in a header file. Upon locating a defined header the connected component label would read and then passed to the deletion functions. This would allow for additional flexibility in the manipulation of cartographic data.



ORIGINAL DATA



RIVER IDENTIFIED FOR
DELETION AND HIGHLIGHTED



RIVER DETECTED BY CHANGING
COLOR VALUE

Figure 7-11. Deletion by Color Change

7.4.4.2 Insert

The creation of new data will modify existing data by the process of replacement. Data insertion can be thought of as overlaying a new feature on the current display, thus masking out any data below it. Performing this function requires some tedious processing steps to maintain data integrity while the insertion is in progress.

To begin, the user would instruct the system to initiate the insertion function. The system's first response would ask the user for a specific feature class, color, or symbol to be inserted. The information provided would tell the system what type of feature will be created. Symbols may take the form of point features such as highway markers or be associated with lineal entities like roads and contours. The procedure for insertion of data will be described first for lineal type features.

Prior to actual data insertion, the user might be given option for viewing displayed data. The user may select to view the entire source image or a subset of it to aid in feature placement. Given the symbol to be used in creating the feature the system would generate a unique component label. Next, a temporary edit file would be created and a pointer set to the first word. The cursor would be enabled and the user instructed to position the cursor. A user directive would then indicate that the insertion of data should begin at the given point.

As the cursor is moved each new pixel location would be read from the display. This coordinate is used to generate the appropriate symbolization. The output from this step would consist of a set of pixel locations to be modified. For example if a railroad was being inserted, the tick marks along the track would be calculated at appropriate intervals. Wherever the tick marks fell, the corresponding pixels will have to be modified.

Each pixel location to be modified would have to be processed individually. First the pixel value memory would be accessed and the value read. This value and pixel coordinate would be written to the edit file, followed by the component label number retrieved from the memory of component labels. From here the new component label would be stored along with a new pixel value in the RAM planes. The pixel location and new value would be sent to the display electronics which would update the screen. When all the pixel locations for the symbolized area have been processed, the next cursor position could be read. This process would continue until the user signaled the end of the insertion. The result would be a display showing the inserted symbolized feature exactly as it would appear on the final output manuscript. In addition, the edit file would contain the necessary information to reconstruct the source image if the user rejected the new feature. If the insertion was accepted and a header file existed then the user would be prompted to enter header data.

It is clear that a tremendous amount of processing would occur when symbolization is applied. This may slow down the systems response to the users' tracing of the feature. If this causes a problem it might be better to first record the cursors movements and apply symbolization afterwards.

Point features would be handled in the same way except only a single cursor position is required. The given symbol would be pulled from the symbol file and generated on the display. Users could also be given the capability to define an area by tracing a line and then filling the area with a pattern and color. Both capabilities would follow the same steps as inserting a lineal feature.

This has been a brief discussion on the complex problem of inserting new data. Conflict resolution is also a major issue when new data must overlay previous information. This topic is covered in Section 7.5.

7.4.4.3 Clip

A clipping function would provide another useful tool in editing and revision of cartographic data. The purpose of such a function is to divide a feature and delete some portion of it. This function could be performed on a local or global basis at the user's option. Local clipping operations would operate on single features, while a global clip could span many different features. Global clips would perform tasks such as sectioning out entire areas of a display. Special purpose charts having only a subset of feature classes could also be lifted out of the current source image. Examples of both types of clipping are shown in Figure 7-12.

To utilize this capability the user would be asked to identify the type of clipping to be performed. When local clipping is desired the user would then be prompted to select the desired features. Global clipping would query the user for the specific feature class. This could include all features or only two; but, must be more than one feature class. From the users response the system would be able to generate a display of the desired features to be clipped.

In both local and global operations, the user would be able to request system generated or user defined clipping lines. The system could generate lines using cursor positions identified by the user. A straight line would be drawn between two points in a contrasting color. This would enable the user to view how the defined line would affect the data. User defined lines would be created by tracking the cursor and recording the screen coordinates. With this capability users could create curved lines and closed polygons for use in specialized clipping operations.

To perform a clip on a single feature the following steps could be used. For the identified feature the system would generate a bounding rectangle (see

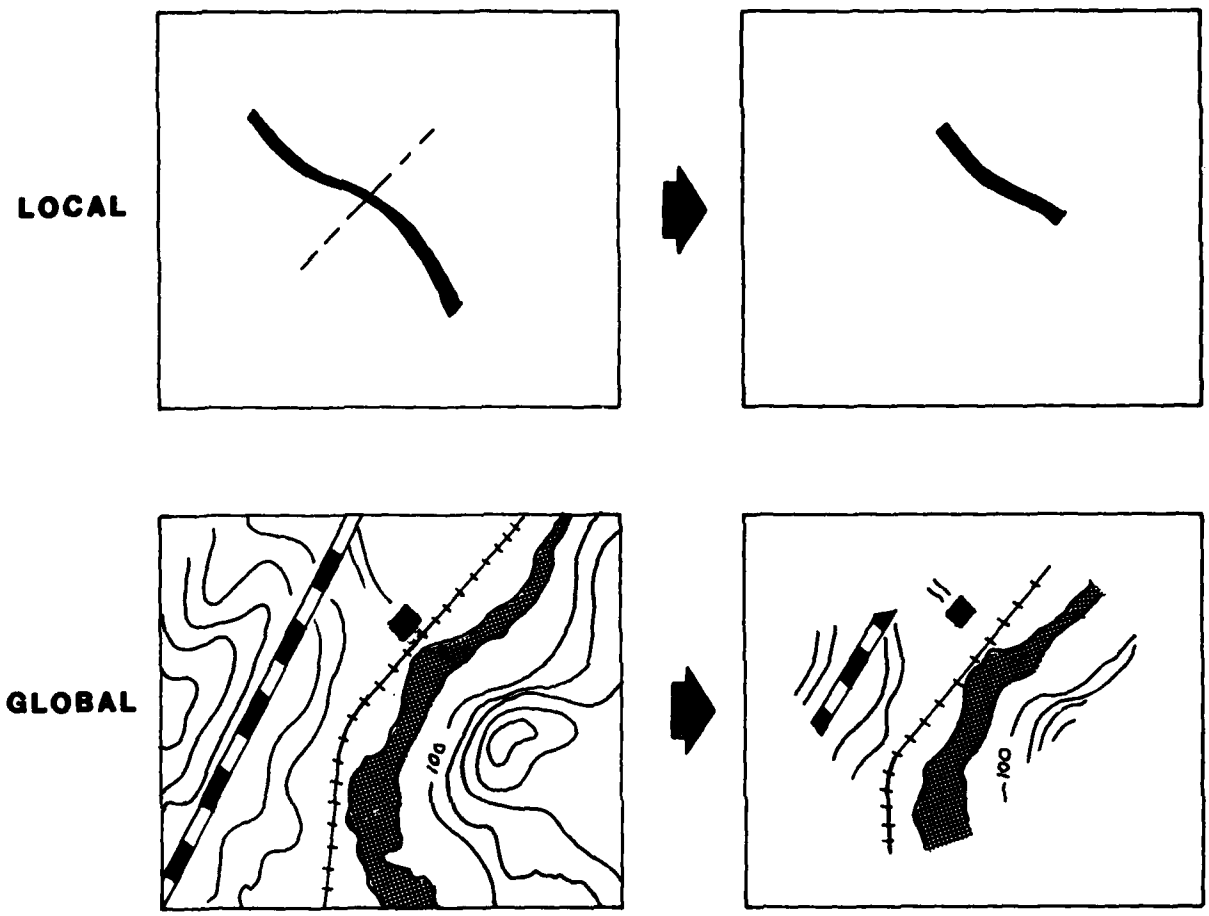


Figure 7-12. Examples of Clipping

Figure 7-13). This rectangle would enclose the feature to limit the amount of required processing. Using two user identified points a clipping line would be generated. The points at which this line bisects the bounding rectangle would be calculated. This would produce vertices for two regular polygons that together equal the bounding rectangle of the feature (See Figure 7-13D). The user would then be queried as to which portion of the feature is to be retained. One method for accomplishing this task is to have the user place the cursor in the area of the clip to be retained. The system would scan the display and read the cursors position. The cursors coordinates would be tested against the vertices of the polygons defined by the clip line. The points defining the polygon containing the cursors position would identify that portion of the feature to be retained.

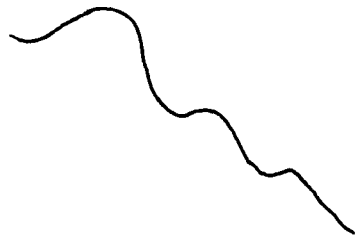
The array of component label numbers would then be scanned. Each location with the identified label would be tested to see if it fell within the correct polygon. A point in polygon technique could be exercised to determine this information. All locations falling within the defined polygon would remain untouched. All points lying outside the polygon would be deleted. The process for deletion would be the same as for deleting an entire feature (described in Section 7.4.4.1).

To perform a global clip on several features a similar process could be implemented. Here the user would identify several features to undergo clipping. Again a bounding rectangle would be generated encompassing all the features if a single line were used. However, the user might also generate a polygon such as shown in Figure 7-14. The area of interest to be retained would be given by the cursor position. A point in polygon technique would then be applied to all selected features. The deletion of undesired areas would be performed just like any other deletion.

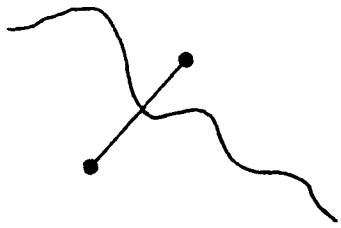
At the end of both a local and global clip, the deleted data will have left a void in the displayed information. Users would be prompted to fill in these areas to complete the clipping process. This step would be handled as if it were the insertion of new data described in the preceding subsection. Finally, to exit from this function users would have to accept or reject the operations performed with the system taking the appropriate actions.

7.4.4.4 Divide

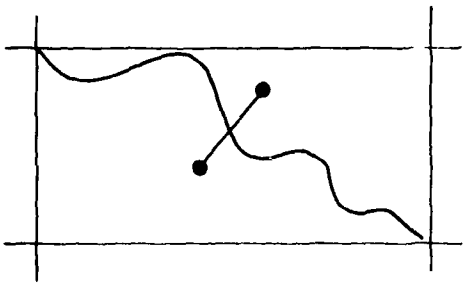
The divide function is almost identical to the clipping function in relation to processing procedures. The difference is that while the divide function also severs features, both portions are retained. The changes occur in the RAM memory planes and may or may not affect the visual output of the data. An example of this function use is in the manipulation of road networks. Intersecting roads may end up with the same connected component labels. Thus any manipulation of even a small part of a road will affect the entire road network. In such cases, it may be desirable to separate part of the road network for easier manipulation.



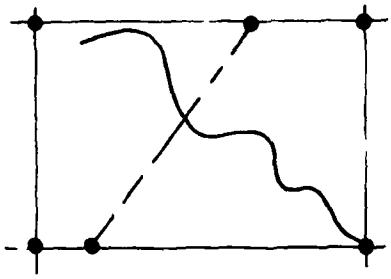
A. IDENTIFY FEATURE



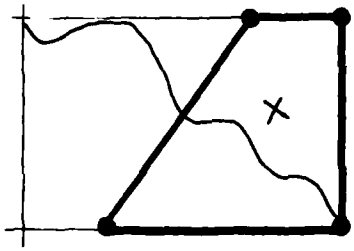
B. DEFINE CLIP LINE



C. GENERATE BOUNDING RECTANGLE



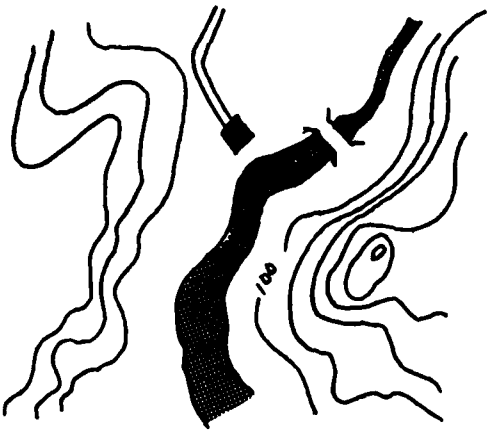
D. GENERATE POLYGON VERTICES



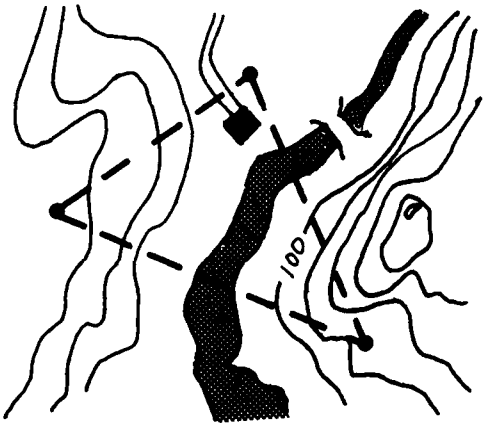
E. IDENTIFY PORTION TO RETAIN

F. DELETE PIXELS NOT FALLING IN SELECTED POLYGON

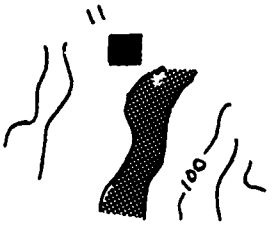
Figure 7-13. Clipping A Feature



A. IDENTIFY FEATURE CLASSES



B. DEFINE A POLYGON



C. DELETE ALL DATA OUTSIDE POLYGON

Figure 7-14. Performing a Global Clip

To perform this, the user would first identify the feature involved and a divide line. The same technique for generating clipping lines could be used here. Likewise the same processing procedures for clipping could be observed for the divide function. In place of the steps which delete the data a new connected component label would be inserted. The result of this operation would leave the display intact with no apparent changes. Therefore, the system should inform the user that the divide has taken place, highlight the new feature, and show the new component label number.

7.4.4.5 Join

A join function would permit multiple like features to be aggregated into a single feature without visual interruption. Features would be combined and gaps between them filled so that a single continuous feature is displayed.

Initiating this function, the user would be called upon to select the features to be joined. Obtaining the component labels for the identified features from the select function, the system would check for compatibility. If the features did not have a like pixel value the user would be notified and the join process terminated. Attempting to join unlike features would only result in erroneous data.

Using the pixel connectivity map, endpoints can be determined. This can easily be envisioned for lineal features, however, regions such as an area of vegetation has no real endpoints. In cases involving such features, the user will be required to define the path between the areas to be joined. The path would then be filled with the pixel value of the two areas to be joined.

With the endpoints derived through the pixel connectivity map, the system can determine what type of join is most logical. Examples of joining lineal features are given in Figure 7-15. The system always uses the endpoints which would create a smooth and short link between the features. The user would always have the final decision and could override the automatic process.

To execute the join the system would use the component label number of the first identified feature as a master label. Thus, no data changes would be made to the first feature. The component label of the second and subsequent features to be joined would be replaced by the master label. Finally, the path or link between features would be generated. An average line width in pixels would be determined from the endpoints of the features. The path between the two features would be modified so that affected pixels would have the same color and component label as the two features. As with all visual modifications, the data changes would be recorded in the edit file. The join process would then terminate after an accept or reject command was given.

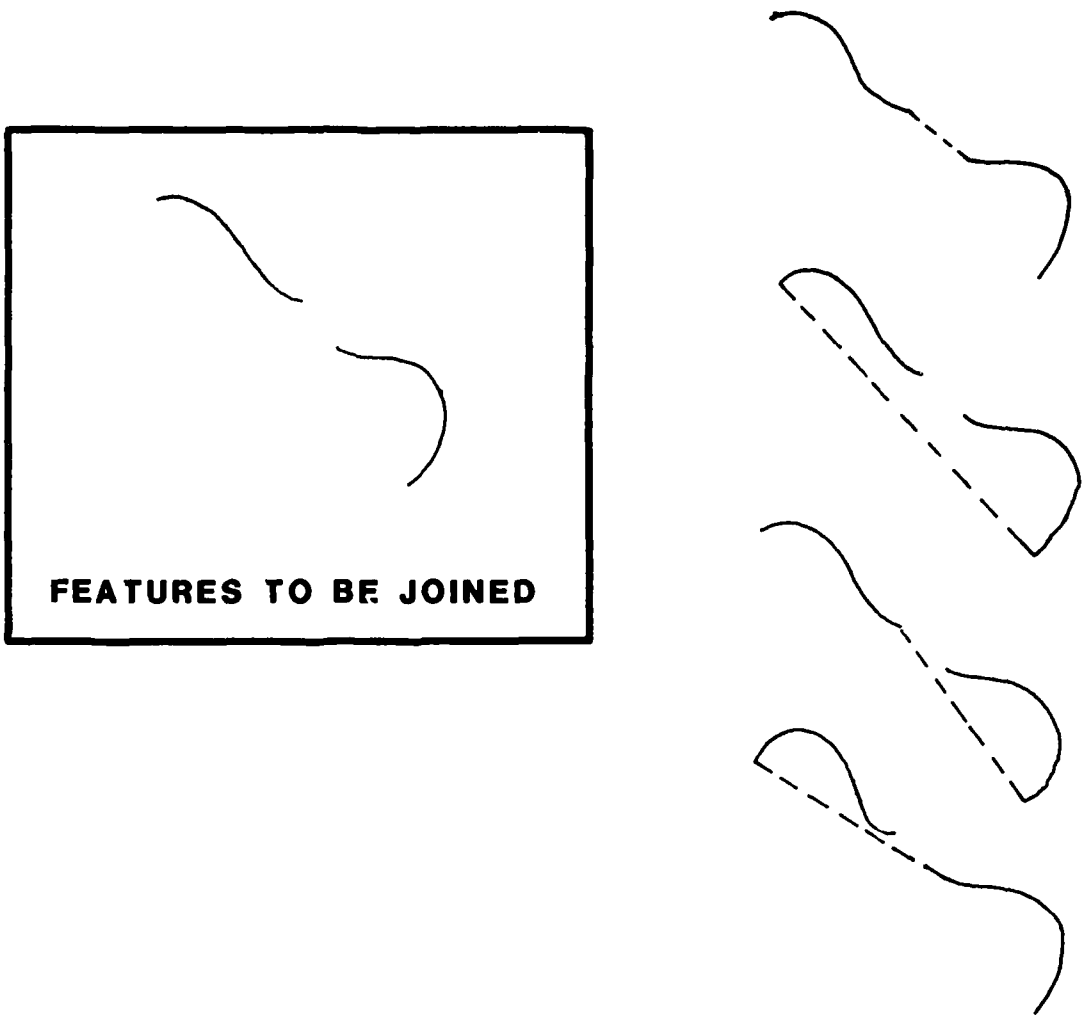


Figure 7-15. Joining Two Features

7.4.4.6 Realign

This function is perhaps the most complex of all the manipulation processes. Here changes to portions of a feature are possible as well as the entire feature. The previous functions discussed operated on the entire features or multiple features. Additionally, each function was a single process, whereas the realignment process includes several different functions. Rotation, translation, scaling, and editing are all part of the realignment function.

Rotation, translation, and scaling of features could be performed by first selecting the feature. The desired transformation would be applied to the feature with all affected pixels recorded in the edit file. In essence the transformations would combine the processes of deletion and insertion with a move operation.

The edit process is more difficult to implement as many operations are required. In addition, most editing will be performed on lineal type features. Point features, such as symbols, will change very little as they are either created, deleted, or transformed to some new position. Regions can be edited by the clip, join, divide functions which can effectively remove or add portions of data. Therefore, editing of lineal features is the primary concern and is used to discuss the possible techniques.

As with all manipulation functions, the first step is to select the feature to be edited. The display functions would then isolate the feature so that only the desired data is seen. Next, two temporary edit files would be created. One to record the modifications and the other to store the affected pixel information. Using the cursor, the user would position to the start point and notify the system via a command. As the cursor is moved from that point, the position is read from screen and stored in an edit file. The pixel location affected and associated data would be stored in the second edit file. The display would be updated at the same time to show the track being created. When the user signaled the end of the edit, the process of recording the cursor position would terminate.

Control would pass to the system which would begin an interpretative process on the data. At this step, the system would assume:

1. The new data recorded will be part of the new feature.

2. Some portion of the original feature is to be retained.

To determine what part of the original feature is to be saved, the endpoints of the new data could be used. If the endpoints both fall on or reasonably near the original feature and are between its endpoints then an internal edit was performed. The system would display the new feature as it

interpretated the edit performed. However, if only one endpoint of the new data is on the feature, then some type of endpoint edit has been done. The system would make an assumption and display the new data with some part of the original feature. In either case, the user would be able to view all possible combinations through interactive commands.

When the correct edit is displayed the user would issue the accept command to initiate the next phase. Here the edit file containing the new data would be read and symbolization applied. The memory of connected component labels and the pixel value map would be adjusted just as if new data were being inserted. The entire source image would be redisplayed reflecting the new data and both edit files destroyed.

7.4.4.7 Tie

A tie function would enable unlike connected components to be linked together without destroying their uniqueness. An example would be the moving of several different features. The user might wish to displace all the road names. Rather than move each individual character in each name, the user could logically link them together and move them all with a single command. In many ways the tie function resembles the join except that each feature retains its identity and no new data is created.

To perform a tie the user would merely identify each feature involved. The connected component labels would be retrieved and saved in a buffer area as equivalency pairs. The effect is the creation of a set of connected components that the system will treat as a single entity. Any further manipulation to any one feature will also have some impact on all the other tied features.

Also required for this function is a capability to untie the logically linked components. This can be achieved by clearing the buffer area or the equivalency list of component labels.

7.4.5 MANIPULATIONS PRODUCING DESCRIPTIVE CHANGES

Cartographic manipulations can also be performed by, and on narrative information if it exists. Thus, when a feature is associated with a header we can identify it through the header number. Functions which operate on the narrative data should provide the capabilities to create, modify, and manipulate it. The following subsections present a few possible functions which might aid in cartographic manipulation tasks.

7.4.5.1 Header Build

Headers offer a way to further characterize a feature beyond symbolization. If this is desired for a raster system, a capability to create unique headers must be provided. Relative to our concept headers will be created only when necessary and using standard conventions. Each feature header would consist of major classes (i.e. roads) and a string of subclasses or attributes (i.e., hard surfaced, single lane).

The header build function would allow for the creation of unique headers and associating them to individual features. The function could be flexible enough to predefine unique headers or build them interactively. For example, a user might build a list of major classes and possibly fill in some or all the attributes for each class. With this list of headers, the user could step through each connected component label number and associate a header with it. Another way is to allow the user to build and associate headers as each connected component label number is displayed. In either situation the features would be displayed on the raster monitor along with the connected component label. When a header number is assigned it is stored along with the connected component label in a header file. Any manipulations which affected the component label numbers would have to take appropriate action to the header file as well.

7.4.5.2 Modify Header

During the course of a manipulation session the user may need to edit or modify a header to reflect a revision to the feature. Such a case is when a road changes from unsurfaced to hard surfaced. The user could produce the modification by identifying the feature or the header number. Once the header number is known, the attributes could be displayed. The users would then identify the fields to change and the values to be entered.

7.4.5.3 Multiple Headering

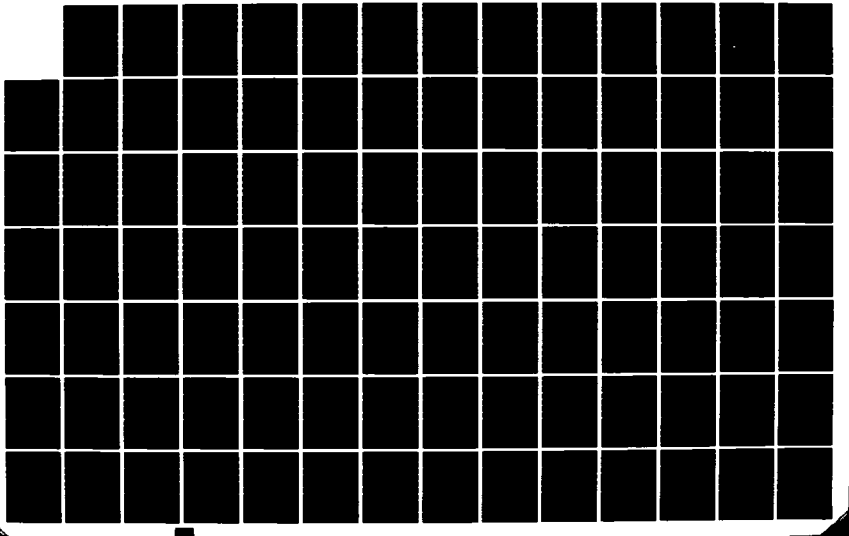
This function would exist as way of associating a single feature to more than one header. The effect would be similar to the tie function in Section 7.4.4.7. Users would identify the feature, indicate a multiple-header operation and add the component label and additional header to the header file. When performing manipulations by header number any feature matching the given header will be affected.

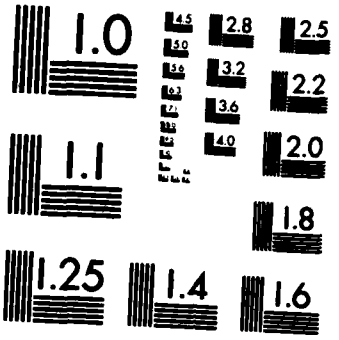
7.4.5.4 Textual Edits

Textual edits could cover a wide range of areas. We refer to it here only as a possible function for creating and editing textual comments or information. Many times during a manipulation session users require an area

AD-A130 195 INTERACTIVE RASTER DATA STRUCTURE STUDY(U) SYNECTICS 3/3
CORP ROME NY D R GALLAURESI ET AL. JAN 83
RADC-TR-83-12 F30602-81-C-0279

UNCLASSIFIED F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

to record notes for further use. A textual file is one possible area to store notes in. Such a function would be implemented by using the systems operating system programs for creation and editing ASCII data.

7.5 Merge and Conflict Resolution

Our concept for an all raster system includes merge and conflict resolution processes as essential components. Although these are discussed in this section, they can not be thought of as separate from the manipulation functions. These processes could be performed along with the cartographic manipulation functions.

Merge is the process of recreating the N x M visual array of data in preparation for output. This is performed at the completion of a manipulation affecting the visual representation of the data. The RAM plane holding the pixel values is the N x M array and is modified to reflect the changes at the end of each manipulation. In this sense, the merge process is automatically performed as a result of a manipulation.

During the manipulation processes there will be countless occurrences of feature coalescence. This is where two or more features share a common location and represent a problem in developing an all raster cartographic system.

When changes are made to existing source materials they may have a wide range of affects. Conflict resolution processes are needed to detect, notify, and correct potential errors during a session. A simple example is the insertion of data which overlap existing data. When a river is being inserted and it crosses a road, feature coalescence will occur. The user could prevent this by stopping the river prior to its intersection of the road and resume insertion on the other side. This is a cumbersome procedures and not always possible. The user may not have a full view of the existing data and therefore would be unaware of underlying features.

A possible solution to this problem is to install procedures to make automatic decisions where possible. The system could make use of a color file which hierarchically store the colors to be manipulated (see Section 7.5). When data is being inserted the pixel value or color is known and can be compared against the value at the location of the insertion. When the new pixel value has a higher priority than the existing value it would replace the current value. New data having a lower priority would be recorded and would not be displayed. Thus pixel denoting a river would not be permitted to replace those depicting a road. The operation would be automatic while the user was tracing a new feature on the display.

In some cases the system might be unable to make a decision as to what has the higher priority. In this situation the user would be prompted for a directive. Another possibility is a user who wishes to perform all conflict resolution tasks manually. Here users would utilize the capability to override the system's automatic processes for conflict resolution.

SECTION 8. SUPPORT ENVIRONMENT

The implementation of a raster cartographic compilation system must include an assessment of the hardware environment which will be supporting both the user and the software. Figure 8-1 presents a set of technologies which must be considered in order to optimize the performance of the overall system. The remainder of this section will present a full system hardware concept and examine technologies to determine their contribution to the support environment. This section is complemented by Appendix C, Relevant Data Storage Technologies which describes appropriate state-of-the-art hardware.

8.1 Hardware Concept

The hardware concept for a raster cartographic compilation system is designed to optimize system resources, incorporate state-of-the-art (SOTA) technology, and provide for distribution of processing tasks wherever possible. With these goals in mind, a description of system hardware concept is in order, (reference Figure 8-2). It is important to note that this hardware concept contains all the elements necessary to support a full raster production system. This enhanced concept could be implemented by providing access to similar devices or capabilities which are available on existing systems. For example, scanned data could be transferred from another raster system. The minimum equipment necessary is identified as the components of the interactive work station.

The host processing units have responsibility for maintenance of the raster data bases, file transfers, coordination of all peripherals, and processing any batch applications tasks. In light of these responsibilitiies, one can see the advantages offered by utilizing two processors; one concerned solely with data base maintenance and file transfer, and the other controlling I/O to and from the peripherals while running applications programs. This is potentially an area where a contribution from a back-end data base processor could benefit the system. The most promising are back-end relational data base machines. The relational data model offers advantages over hierarchic, network, and semantic models for maintaining large raster files by offering:

1. Content addressability
2. High degree of parallelism
3. Very robust and flexible

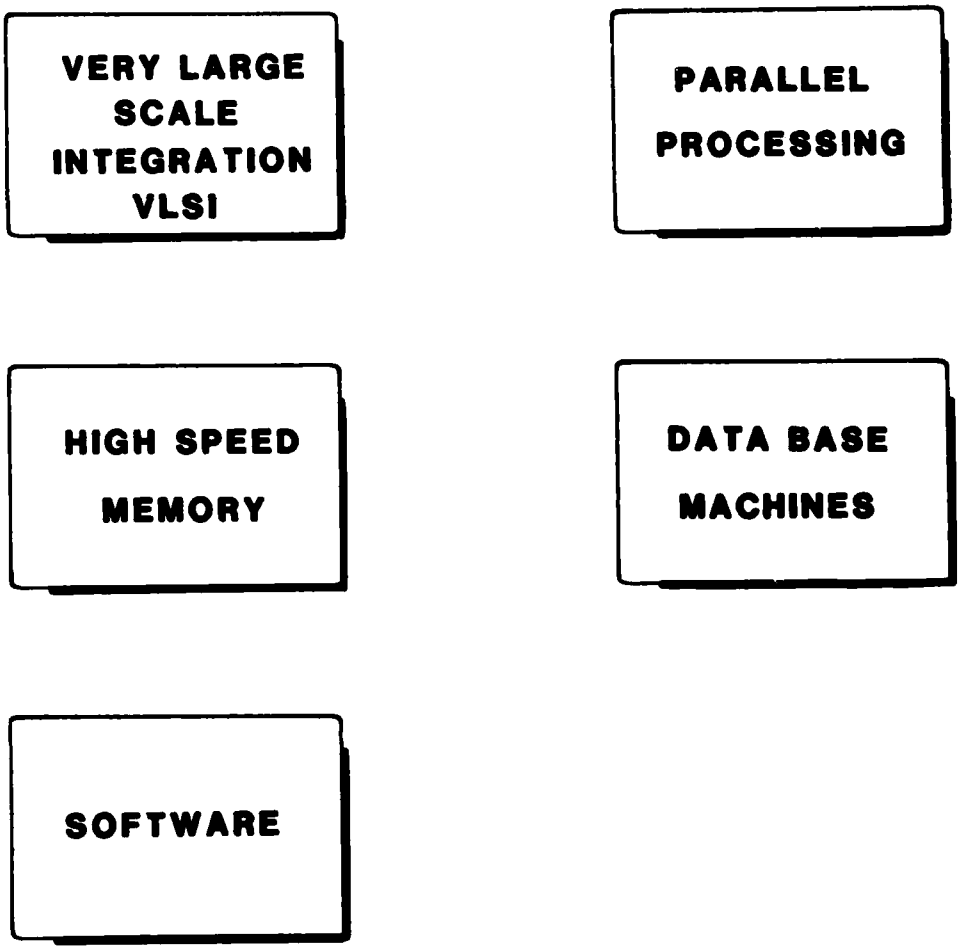


Figure 8-1. Emerging Technologies for Consideration

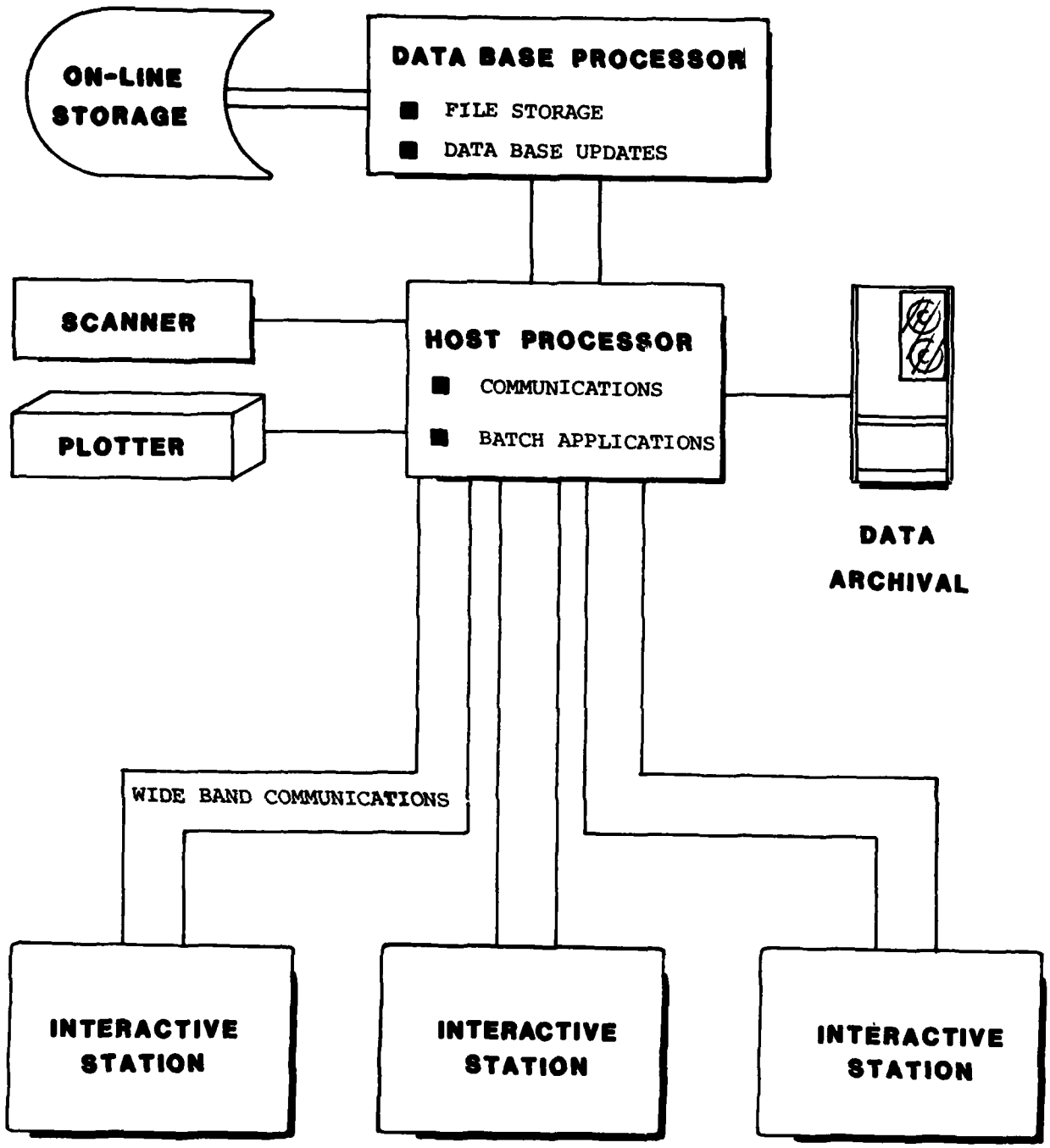


Figure 8-2. IRDS System Hardware Concept

4. Simple structure

5. Ease of defining the data base (files).

The idea of a back-end processor is to off-load the data management functions from the host onto a special purpose minicomputer. This frees up the host machine for running application programs and controlling other peripherals. A discussion of the state-of-the-art technology in back-end data base machines is included in Appendix C, Relevant Data Storage Technologies.

The front-end host processor has responsibility for:

1. Communicating data requests in the proper format to the back-end processor.

2. Coordinating use of all peripherals.

3. Execution of batch application programs such as:

   a. file reformatting

   b. project transformations

   c. file backup

   d. data compression/decompression

In light of these responsibilities the host processor should be large enough to support both background applications programs and be responsive to the data requests from the interactive stations.

A raster scanner is used as a sharable device for inputting both multicolor, large format maps and monochrome imagery.

The data archival component is used for file backup and temporary off-line storage. Since the purpose of the system is product compilation, long term data storage should not be a requirement of the system. Traditionally, data archival has been performed by magnetic tape units, which also offer flexibility as an interface for data transport. Magnetic tape technology is described in Appendix C.

A sharable raster plotter capability is intended primarily for hardcopy examination, although its role could be as a finishing output device as well.

On-line data storage requirements of a raster compilation system are massive and would require SOTA in high density storage (Figure 8-3).

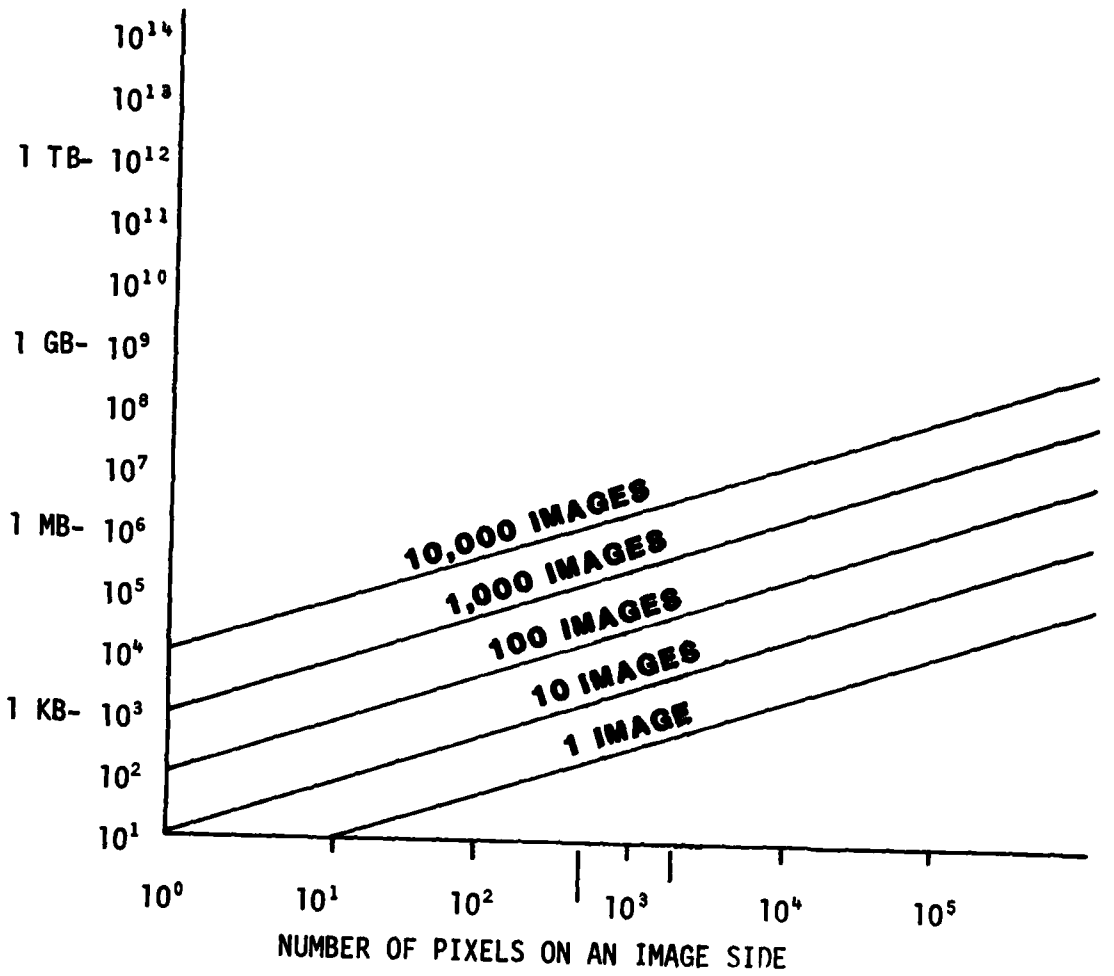


Figure 8-3. Imagery Data Storage Requirements

Data volumes for a raster image are typically 10^3 times as large as data volume for vector storage normally encountered in non-pictorial applications (Danielsson, 1980). Figure 8-4 presents a worst case storage scenario for one raster scanned source material. Assuming the minimum scanning resolution (4 mils) for the largest size source material (40"x60") and storing each pixel as a unique 8-bit byte, a raster image requires 150 megabytes of on-line storage. Using run length encoding as a simple data compression technique can reduce the required storage by 30-90% depending upon the nature of the source material. It is obvious that a data storage goal should be to limit the number of redundant or similar files. This requirement also is applicable to the interactive station configuration. Until optical disk technology overcomes the current problems with high error rates and can offer read/write/rewrite capabilities, magnetic medium remains recommended for high capacity, reliable on-line storage (reference Appendix C).

Several interactive stations are connected to the host processor(s) via wideband communications lines. The wideband communications are necessary in order to accommodate the high volume, rapid response data transfers. Assuming imagery is stored and transmitted with 256 color codes using 8-bits per pixel, (no data compaction, padding, error detection, nor parity bits) transmission rates must meet or exceed 50 kilobauds as shown in Figure 8-5.

Each interactive station is configured according to the diagram in Figure 8-6. The display monitor should provide 256 displayable colors and 1024x1024 viewable resolution.

Interaction with features displayed on the monitor is via both a standard cursor control device such as a graphic tablet and an alphanumeric keyboard with programmable function keys. Wise use of the function keys would facilitate specification and execution of many manipulation and display functions. This is especially critical in our proposed editing environment where many assumptions, decisions, and possibilities are made by the software functions with guidance, checkpointing acknowledgement and approval coming from the operator. The function keys can designate not only the software functions but also can be used to provide yes/no answers and indication of particular significant points that should be searched for.

The video electronics component of the station controls display refresh, color, and gray scale selection, highlighting and blinking, and hardware implementation of common display functions such as translation and scaling. These attributes are common in commerically available display controllers.

Random access memory (RAM) memory planes contain the data elements (i.e., Pixel Value, PCM, and Label) described in Section 6, in 8-bit planes. These elements would require a minimum of three planes. State-of-the-art in RAM technology is described in Appendix C and applicable memory configuration descriptions can be gathered from commercially available display hardware such as the Gould/DeAnza IP8500. Ideally, the RAM memory configuration

DATA FILE REQUIREMENTS

SOURCE MATERIAL SIZE

40" BY 60" = 2400 SQUARE INCHES

MINIMUM SCANNING RESOLUTION

4 MIL = 250 PIXELS PER INCH

= 62,500 PIXELS PER SQUARE INCH

NUMBER OF PIXELS IN SOURCE MATERIAL

2400 * 62500 = 1.5 E^8

DATA STORAGE ASSUMING 8-BIT BYTES

1.5 E08 BYTES OR 150 MEGABYTES

Figure 8-4. Worst Case Disk Space Requirements

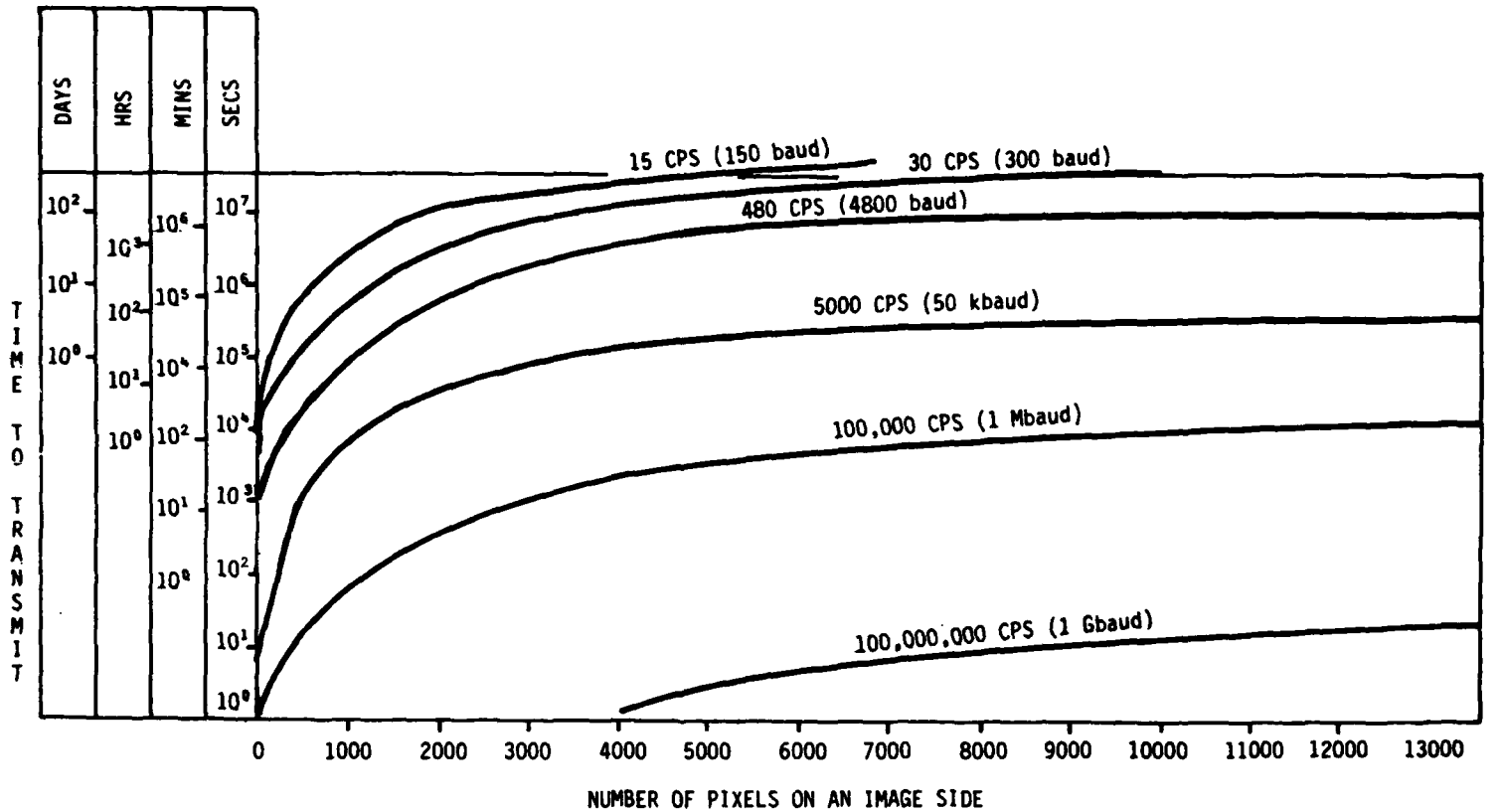


Figure 8-5. Soft Copy Imagery Transmission Rates Versus Transmission Time

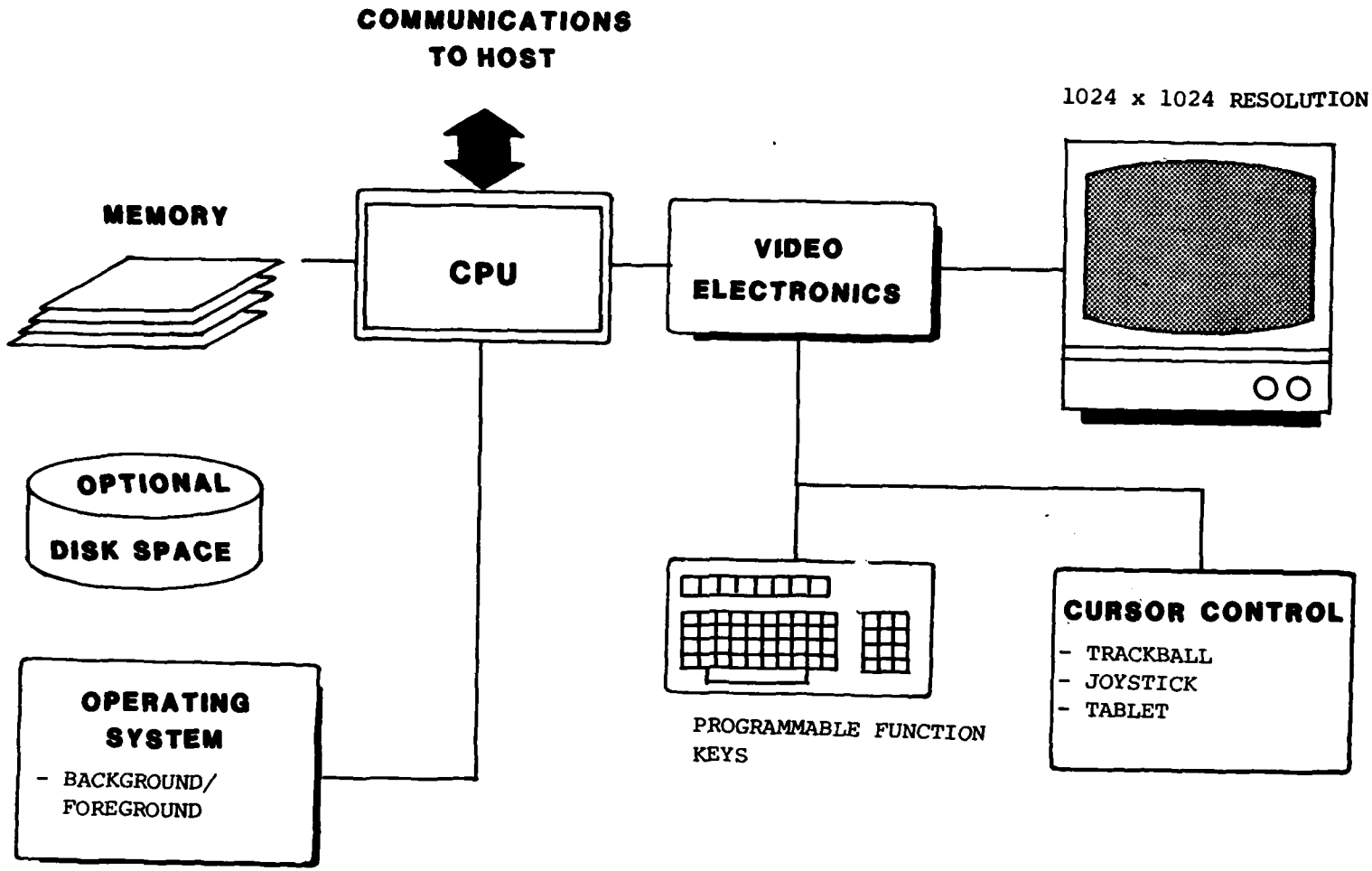


Figure 8-6. Interactive Station Hardware Concept

would be larger than the displayable memory, such as 2048^2 or 4096^2, in order to support scrolling through the image. This would allow instant access to a sizable portion of a file without requiring additional data be transferred from disk storage.

Use of disk space for the interactive station offers three possible scenarios:

1. The local disk space for the work station would contain a full copy of the entire source file to be reviewed and edited.

2. The local disk space would be used similar to a cache memory and contain enough data to represent a sizable portion of the file.

3. There would be no local disk space available locally and the data to be edited would be stored in RAM memory.

These scenarios offer tradeoffs in speed, cost, and flexibility. Scenario 1 offers rapid access to an entire map/chart, but at the cost of needing a very large disk (Figure 8-4). Scenario 2 offers a compromise, but necessitates more complex software for coordinating files. The third scenario is attractive due to simplicity and low cost if data transfer requests are honored rapidly from the host and the review/edit process does not require jumping around several areas of a map.

The processing unit of the station would not only include a typical serial processor for execution of sequential tasks but also special function parallel processors for rapid execution of parallel operations. A parallel processing unit would be invoked to perform a specific task in a fashion similar to a subroutine call from a main program. The following, Section 8.2 discusses promising parallel processing architecture.

8.2 Parallel Architectures

Advances in the design and fabrication of VLSI circuits ensure the feasibility of implementation of high performance special-purpose computers consisting of highly parallel computing elements on the order of 10^4 to 10^5 in number. Highly parallel structures promise tremendous speed improvements over the fastest of conventional modern machines.

Highly parallel architectures have structural properties that make the application of VLSI look very promising (Fairburn). By the virtue of the repetition of basic computational elements and communications with only their nearest neighbors, parallel systems stand to gain the most from VLSI. For added measure, VLSI offers very fast and inexpensive computational elements including bidirectional transmission gates that permit shifter arrays to be configured in very compact and inexpensive NMOS chips.

The infancy of the new VLSI technology has already fostered a new generation of computer science professionals composed of VLSI designers, numerical analysis experts, software engineers, and application area specialists.

The algorithms and data structures that are the essential fabric of the conceptual cartographic raster processing system appear to be ideally suited for highly parallel computation. Consequently the emergence of VLSI technology and its applicability to highly parallel structures demanded investigation by the IRDS research team. This information was prepared to capture the essence of the SOTA of VLSI relative to candidate parallel architectures for the conceptual raster system.

8.2.1 HIGHLY PARALLEL STRUCTURES

A highly parallel structure has three main characteristics:

1. It is composed of a large number of possibly heterogeneous computing elements.

2. The number of these elements is conceptually expandable at a cost not much greater than linear, achieving a speedup that is not much lower than linear.

3. It is used to solve one single problem at a time.

Given our conceptual raster processing system, we wish to match an optimal parallel architecture with consideration to cost-benefit tradeoffs between special purpose vs. general purpose structures. We also wish to

know the most efficient and cost effective interconnection network among the processors. Finally, effective methods for algorithm design and programming on such a highly parallel system must be attained.

Highly parallel architectures can be categorized in the following general classes: multiple-special purpose functional units, associative processors, array processors, data flow processors, functional programming language processors, and multiple CPUs. Of these classes, the multiple-special purpose functional units are the fastest and can be designed to achieve nearly 100 percent hardware efficiency (Haynes, 1982). In addition they require little or no software, but they also imply an attendant loss in generality. Multiple special purpose functional units are most cost-effectively applied to performing computer-bound operations that constitute the computational intensive kernel of a processing problem.

Systolic array architectures are an example of multiple special purpose functional units that possess fixed interconnections and are capable of extremely high speed and efficiency, but at the expense of performing only a few operations.

Some examples of systolic designs that have been successfully implemented to improve computer bound computations include:

1. Discrete Fourier transform
2. 1D and 2D median filtering
3. Geometric warping
4. 1D and 2D convolution and correlation
5. Interpolation
6. Matrix arithmetic
7. Simultaneous linear solutions
8. Relational data base operations
9. Pattern matching
10. Encoding (polynomial division)
11. Searching algorithms
12. Data Structures

Systolic designs may be applied to any computer bound problem that is regular (i.e., where repetitive computations are performed on a large set of data).

Systolic arrays are designed to achieve optimal processor interconnection communications that are tailored to a specific problem/algorithm combination. Interconnection of processors within a systolic array are characterized by low hardware complexity/cost, very high efficiency of processor utilization and low degree of generality due to the fixed geometry of the interconnections. At the price of specialization, they provide essentially optimal solutions to the highly parallel architecture design problem.

Systolic arrays utilize nearest-neighbor processor connections in which the communications and processors are optimized for specific problem classes. I/O is overlapped with computation, in which each operand is input only once and operated upon many times. The bandwidth of the processor's computation capability must be appropriately matched with the I/O capacity of the bus furnishing the data to the systolic array. Therefore, the ratio of input operations to arithmetic operations is an important metric. The ultimate performance goal of the systolic array is a computation rate that balances the I/O bandwidth of its supporting host processor.

8.2.2 SYSTOLIC ARCHITECTURES

A systolic system consists of a set of interconnected cells, each capable of performing some simple operation (Kung, 1982). Cells in a systolic system are interconnected in an array or tree. Information flows between cells in a piplined fashion and only cells on the array boundaries may be used as I/O ports for the system. The systolic approach is often a relatively simple and inexpensive matter in order to speed up a compute-bound computation (i.e., a computation in which the total number of computation operations exceeds the total number of input/output operations).

The basic principle of a systolic system (Figure 8-7) is to replace a single processing element with an array of processing elements or cells and thereby achieve higher computational throughput. The systolic approach fetches an operand from memory and passes it along from cell to cell along the array, where multiple operations are performed on it in a repetitive manner.

The systolic approach was an architectual concept originally proposed for VLSI implementation of matrix operations. A typical example of a wideclass of computations suitable to systolic designs is the convolution computation which is common to applications such as filtering, pattern matching, correlation, interpolation, fast Fourier transforms, polynomial evaluation, and polynomial multiplication and division. In general, the convolution problem is that of combining two data streams, 1) a sequence of weights and 2) an input sequence to compute a result sequence. Each input, X_i, is multiplied by each of the K weights as follows:

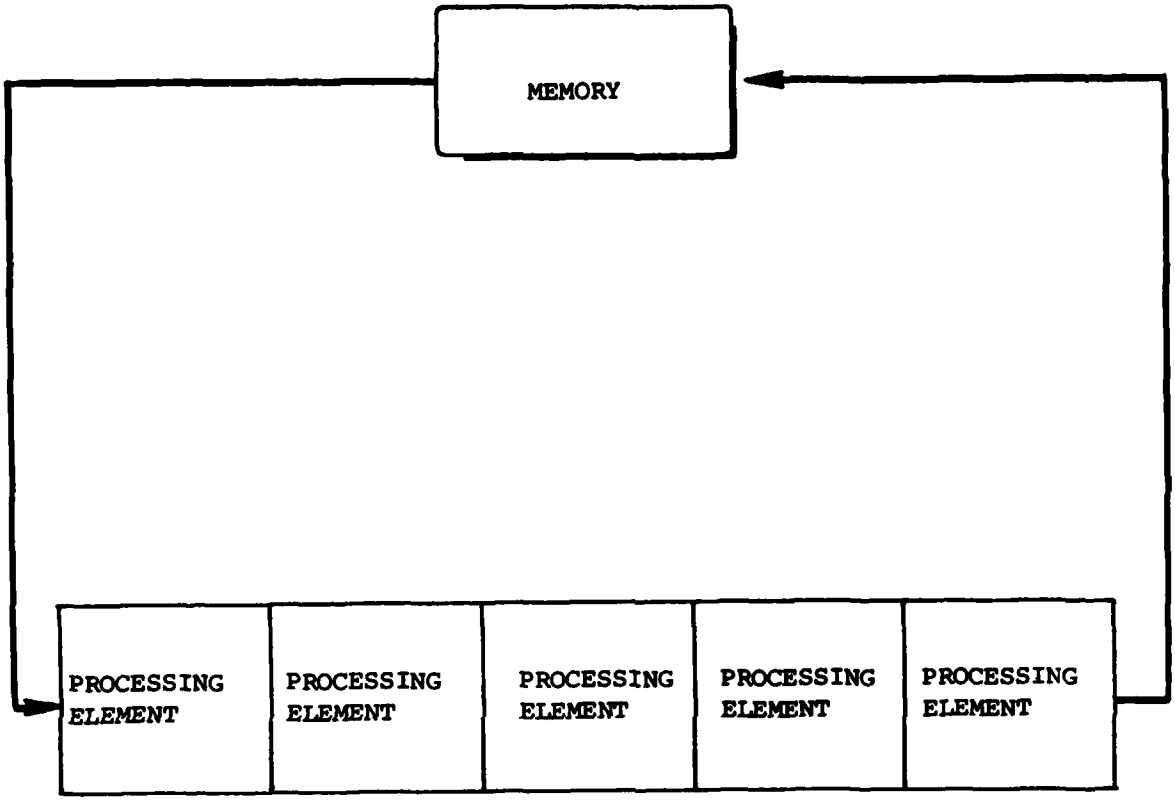


Figure 8-7. Basic Principle of Systolic System

$$Y_i = W_1X_i + W_2X_{i+1} + \ldots + W_kX_{i+k-1}$$

If multiplication and addition are substituted by comparison and boolean AND, respectively the convolution reduces to the pattern matching algorithm.

In our conceptual raster system pattern matching is an integral part of generating pixel connectivity map arrays for connected components and for detecting edit-significant external points by the template matching process. Consequently the systolic approach for the convolution problem is highly applicable to the IRDS investigations.

8.2.2.1 Systolic Array Example as Applied to Pixel Connectivity Map Encoding

The pipelined neighborhood connectivity encoder described by Sobel has an analogue in the realm of simple systolic convolution arrays. In our systolic approach we design the encoder to broadcast the class/label code of the center pixel of a 3x3 neighborhood (Reference Figure 8-8) to each of nine simple cells. Each cell in the systolic array performs the function of comparison and depending on the result sets a status indicator bit. A fan-in technique in which all of the status indicators are collected from the cells and assembled into an 8-bit Pixel Connectivity Map register is performed. The pixel connectivity map register is then moved to an output array at a relative address location corresponding to the center pixel.

The high level diagram of the systolic array for encoding the pixel connectivity map is provided in Figure 8-9. In this example, the systolic array passes over the segmented and labeled raster data in the same sequence as the original raster scan. The assembled pixel connectivity map is output to array C in the same order as the scan sequence.

On the next cycle the class/label code of pixel, Pi-N, is broadcast to each of the nine registers. At the same time the Pj's move systolically from cell to cell in the right to left direction, that is, each of them moves over the cell to its left during each cycle. Simultaneously, three new pixels (Pi-2N+1, Pi-N+1, Pi+1) are loaded into the three right most cells.

Providing and collecting the data to/from the cells during each cycle requires the use of a bus or tree-like network. Consequently the PCM encoder is categorized as a semi-systolic array, because it does require global data communications. Pure systolic arrays do not require global data communication and therefore have much greater potential for extended growth in number of cells without encountering synchronization problems.

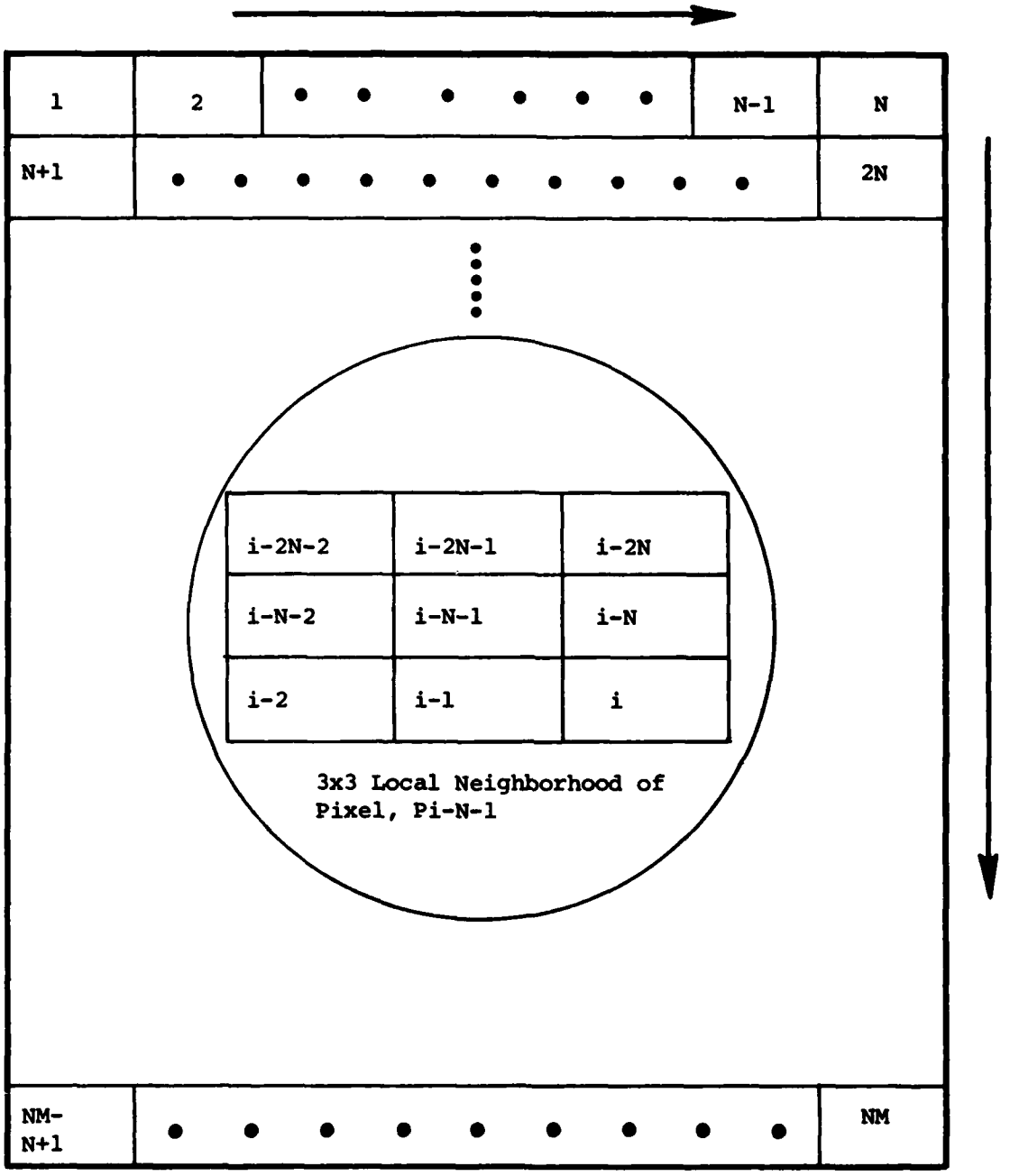


Figure 8-8. Pixel Addresses in Raster Scan Sequence of NxM Image Array

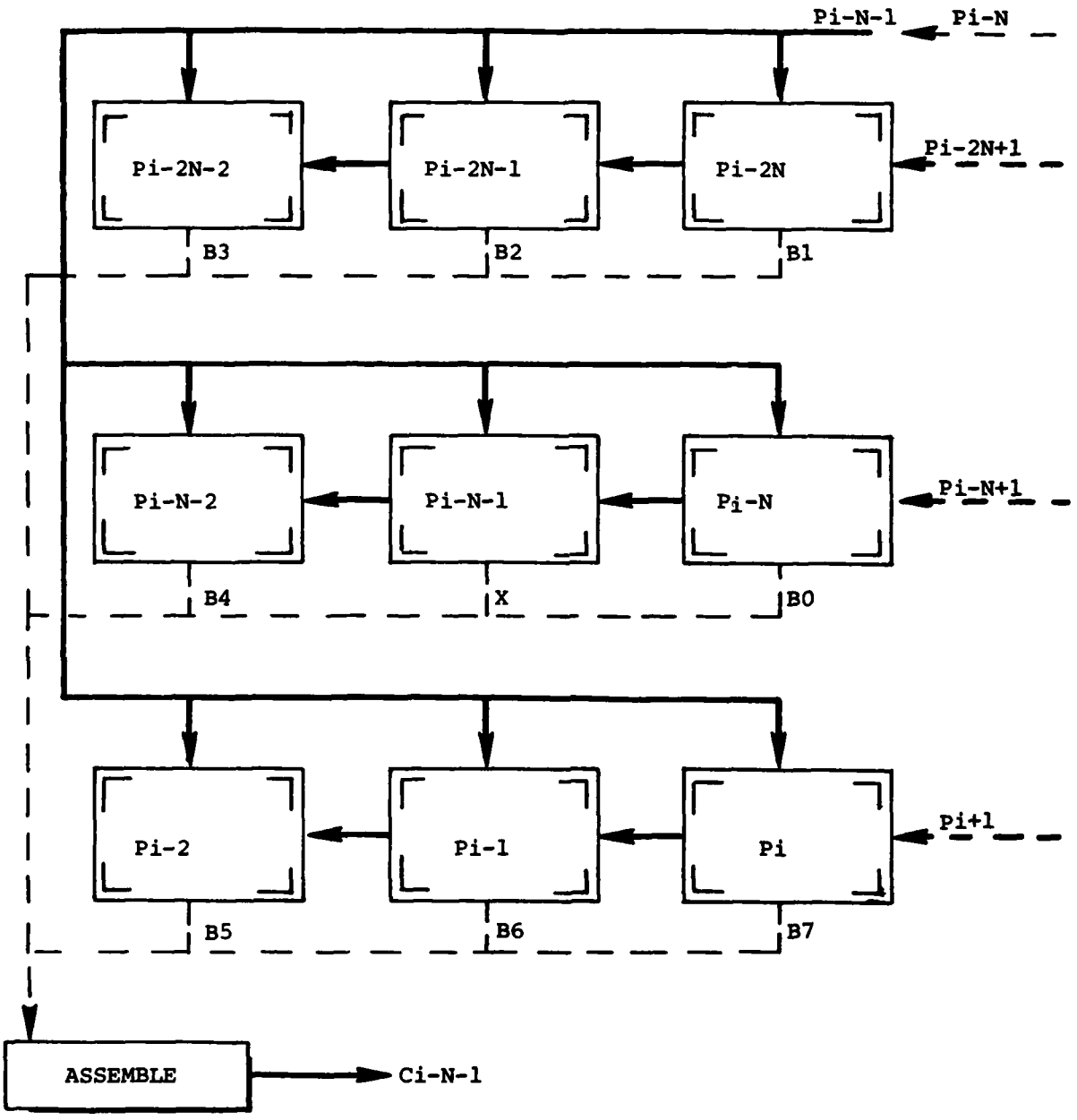


Figure 8-9. Systolic Array for Encoding Pixel Connectivity Map

8.2.2.2 Using Systolic Arrays for Template Matching in Detecting Local Maxima and Minima of External and Hole Contours

Pattern matching the 24 templates with the pixel connectivity map array to determine the significant edit points can be a relatively slow process with conventional Von Neumann machine architectures. The systolic approach has high potential payoff in terms of faster response time as applied to this pattern matching problem.

The pure systolic convolution array proposed for the pattern matching solution is shown in Figure 8-10.

Each cell is preloaded with a unique template, T_k and a companion bit mask, M_k which remain in the cell throughout the computation. On each cycle the next pixel connectivity map C_i is fetched from memory and loaded into the leftmost cell. The pixel connectivity map data moves from cell to cell in left to right fashion. Each cell performs a logical mask and comparison operation upon the pixel connectivity map operand. If a high is found, a max/min type bit is inclusively "OR"ed into a partial output register. The output register associated with the leftmost cell is preinitialized to zero at the beginning of each cycle prior to the computation. The partial output registers, t_j's are moved along in synchronization with the cell-to-cell moves of the pixel connectivity map. Alternatively, the type code could be an extended field associated with the pixel connectivity map and be passed along as a unit within the C_j's. The output at the rightmost cell is accordingly returned to memory. Note that a comparison is made for each C_j throughout all 24 cells. This design utilizes more concurrency than in our first example by virtue of the greater number of systolic cells utilized.

These two examples are just a few of the compute bound regular computations that can benefit from the systolic approach within the conceptual cartographic raster system.

8.2.3 CRITERIA FOR THE DESIGN OF SYSTOLIC ARRAYS

The advantages of the systolic approach are summarized herein:

1. The design makes multiple use of each input data item. It is strongly recommended that each input data element travel through the entire array of cells so that it is used at each cell.

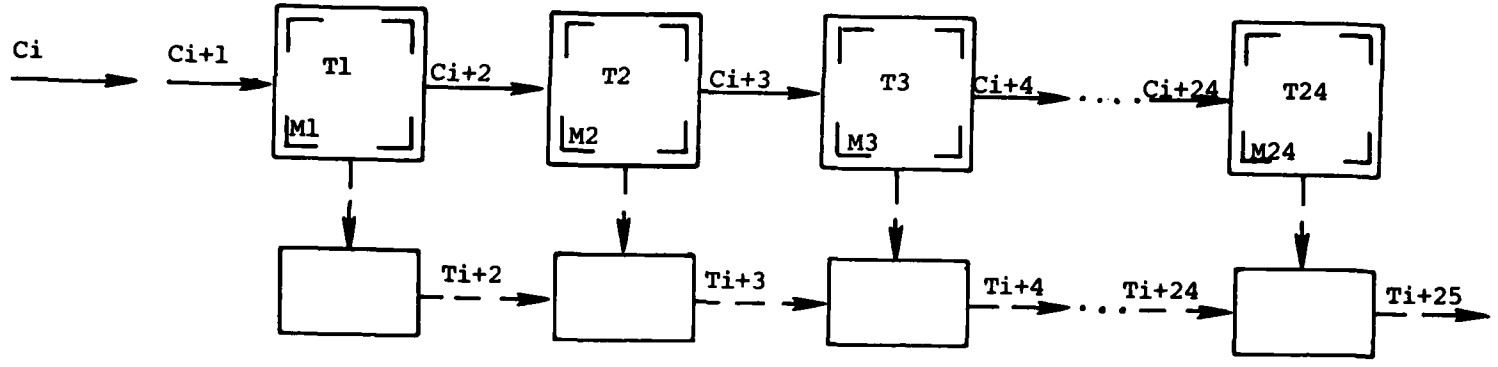


Figure 8-10. Systolic Array For Detecting Local Maxima and Minima Points

2. The design uses extensive concurrency of many simple cells rather than the sequential use of a few powerful processors. In practice systolic arrays can be chained together to form very powerful systems. Nested pipelining will become increasingly commonplace as VLSI makes staging more affordable.

3. There should be only a few types of simple cells to curtail design and implementation costs.

4. Data and Control Flows are simple and regular. Systolic designs are inherently modular and expandable and present no difficult synchronization or resource conflict problems.

In summary, these design criteria and advantages of systolic design yield high performance parallel computing architectures. A unique characteristic of the systolic approach is that as the number of cells expands both the system cost and performance increase proportionally.

8.2.4 SUMMARY

The advent of VLSI promises that a powerful parallel machine architecture can be fabricated to support an affordable cartographic raster processing system. A fundamental system architecture would involve a fast minicomputer connected to a fast bus (or busses). Conceptually these special purpose devices would functionally serve in a manner analogous to subroutines in a software library. The utility of these special purpose devices will depend heavily on the bandwidth of the bus, which must be great enough to furnish data at a sufficiently high rate such that the devices may achieve high processing efficiency.

The goal is the effective use of these special purpose devices such as systolic processors to offload the system with regular compute bound raster processing computations. To achieve this goal further research is needed in order to specify building blocks for systolic arrays such that they can be programmed to accommodate a variety of different raster processing algorithms with little effort. In addition, a system integration design analysis for providing a convenient means of incorporating these special purpose devices and understanding their effective utilization with a cartographic raster system should be conducted.

SECTION 9. RECOMMENDATIONS AND CONCLUSIONS

The material presented in the previous sections is synopsized in a summary graphic, Figure 9-1. This final technical report has presented a comprehensive survey of raster data structures from many applications, presented a description of the hardware and software components of a raster compilation system and presented a beginning for a system design plan encompassing the raster editing software, parallel processing techniques and parallel hardware architecture. What remains to be accomplished is the implementation and testing of the presented techniques in a laboratory environment. From this report, the next logical step is the preparation of a more detailed functional software description, module description in a program design language, and finally, coding and testing alternative techniques to ensure efficiency.

The results of the IRDS study also have benefits outside of the cartographic compilation environment. Potential spinoffs include:

1. The suitability of the Pixel Connectivity Map (PCM) technique for other templating, feature tracking, and automated registration applications. By using PCM bit mask templates significant image point can be classified to assist these locational dependent functions.

2. The survey of raster data structures identified many raster data structures which are suited for other applications, such as spatial data management and pattern recognition.

3. Appendix B presents a data structure and algorithm evaluation technique which can be applied to a wide range of data structures.

4. Finally, in the course of the study there has been an examination and assessment of the state-of-the-art in raster technologies.

- FINAL IRDS TECHNICAL REPORT
  - COMPREHENSIVE SURVEY OF RASTER DATA STRUCTURES
  - DESCRIBES COMPONENTS OF A RASTER COMPILATION SYSTEM
  - DESIGN PLAN FOR:
    - RASTER EDITING SOFTWARE
    - PARALLEL PROCESSING TECHNIQUES
    - PARALLEL HARDWARE ARCHITECTURES

- APPLICABILITY TO OTHER AREAS:
  - PCM TECHNIQUE IS SUITED FOR TEMPLATING, FEATURE RECOGNITION, FEATURE TRACKING, AUTO-REGISTRATION EFFORTS
  - RASTER DATA STRUCTURES IDENTIFIED FOR OTHER APPLICATIONS (RASTER DATA BASE, PATTERN RECOGNITION)
  - DATA STRUCTURE EVALUATION CRITERIA
  - RASTER TECHNOLOGY ASSESSMENT

Figure 9-1. Interactive Raster Data Structures Summary

REFERENCES

Aoki, M. "Rectangular Region Coding for Image Data Compression." Pattern Recognition II (May 1979), 297-312.

Alexandridis and Kinger, "Picture Decomposition, Tree Data-Structures and Identifying Directional Symmetries as Node Combinations." Computer Graphics and Image Processing 8 (1978), 43-77.

Barr, Avron and Fergenbaum, Edward A. The Handbook of Artificial Intelligence Volume I. William Kaufmann, Inc., 1981.

Baudelair, P. and Stome, M. "Techniques for Interactive Raster Graphics." Computer Graphics 14 (July 1980). 314-320.

Berztiss, A. Data Structures 2nd ed. Academic Press, New York, 1975.

Bouille, F. "Structuring Cartographic and Spatial Processes with the Hypergraph Based Data Structure." Universite CURIE France.

Bracken, P. A., Van Wie, P. H., and Dalton, J. T. "Computer Mapping Software on the AOIPS." Harvard Library of Computer Graphics, 1979 Mapping Collection, Volume 2, pp. 31-41.

Brown, D. C. and Chandrasekuran, B. "Design Considerations for Picture Production in a Natural Language Graphics System." Computer Graphics, Volume 15, No. 2., 1981.

Browning and Tanimoto, "Segementation of Pictures into Regions with a Tile-by-Tile Method." Pattern Recognition 15, 1 (1982), 1-10.

Bryant, N. A. and Zobrist, A. L. "An Image Based Information System: Architecture for Integrating Satellite Imagery and Cartographic Data Bases." AUTO CARTO IV, 1980, pp. 43-48.

Cayley, A. "On the Analytical Form Called Trees." Amer Math J. (1981) 266-268.

Cederberg, R. "A Data Structure for a Raster Map Data Base." AUTO CARTO IV, Volume 2, 1979, pp. 85-92.

Cederberg, R. On the Coding, Processing and Display of Binary Images, 1980 LTAB, Linkoping, Sweden.

_____. "Chain-Link Coding and segmentation for Raster-Scan Devices." Computer Graphics and Image Processing, 1979, pp. 224-244.

REFERENCES (Continued)

Chang, N. S., and Fu, K. S., "Picture Query Languages for Pictorial Data Base Systems." IEEE Computer, November 1981.

Chang, N. S., and Fu, K. S., "A Query Language for Relational Image Data Base Systems." Proceedings of IEEE Workshop Picture Data Description and Management, 1980.

Chang, S. K., Donato, N., McCormick, B. H., Reuss, R., and Roichett, R., "A Relational Data Base System for Pictures." Proceedings of Workshop on Picture Data Description and Management, IEEE Society, 1977.

Chang, S. K., and Kunii, T. L., "Pictorial Data Base Systems." IEEE Computer November 1981

Chen, P. P., "The Entity-Relationship Model-Toward a Unified View of Data." ACM Transactions on Data Base Systems, Volume 1, pp. 9-36.

Cushing and Varna, "Foundation of a Knowledge Representation System for Image Understanding." Naval Electronic System Command, (1980).

Dalton, J. T., Billingsley, J. B., "Interactive Mapping Software of the Domestic Information Display System." AUTO CARTO IV, Volume 1, 1979, pp. 119-126.

Dalton, J. T., Winkert, G. E., and Quann, J. J., "A Raster-Encoded Polygon Data Structure for Interactive Mapping." Harvard Library of Computer Graphics, 1980 Mapping Collection, Volume 8, pp. 141-158.

Danielsson, P. E., Kruse, B., and Gudmundson, B., "Memory Hierarchies in PICAP II", Proceedings of the Workshop on Picture Data Description and Management, IEEE Computer Society, 1980, pp. 275-280.

Derrenbacher, R. L., Lazzara, A. V., Lubbes, R. K., and Virkler, G. W., "Raster-Lineal Editing Software." Final Report, RADC-TR-74-248, 1974.

Derrenbacher, R. L., Nasci, R. A., and Winter, W. H., "Raster-Lineal Conversion Analysis Project." Final Technical Report, RADC-TR-77-421, 1977.

Doctor and Torber, "Display Techniques for Oct-Tree-Encoded Object." IEEE Computer Graphics and Applications 1, 7, (1981), 29-38.

Dyer, C. R., and Rosenfeld, A., "Propagation Algorithms for Framing Rectangle Construction", Pattern Recognition, Vol. 12, pp. 211-215, 1980, Pergamon Press, Ltd.

REFERENCES (Continued)

Dougenick, J., "Whirlpool: A Geometric Processor for Polygon Coverage Data." AUTO CARTO IV, Volume II, 1979, pp. 304-311.

Dueker, K. J., "Land Resource Information Systems: Spatial and Attribute Resolution Issues." AUTO CARTO IV, Volume 2, 1979, pp. 328-336.

Dyer, C., "Space Efficiency of Region Representation by Quad Trees." IEEE (1980).

Estes, J. E., and Whitman, R. I., "Remote Sensor Inputs to Information Systems Research at the National Aeronautics and Space Administration." Harvard Library of Computer Graphics, 1980 Mapping Collection, Volume 10.

Euler, L., "Solutio Problematis and geomitriam situs pertinentis." Comment Acadamiae Sci I Petropolitanae (1736) 128-140.

Even, S., Graph Algorithms. Computer Science Press, Potomac, MA, 1979.

Freeman, H., "The Generalized Chain Code for Map Data Encoding and Processing." Air Force Offices of Scientific Research, 1978.

Freeman, H., "Application of the Generalized Chain Coding Scheme to Map Data Processing," Proceedings of IEEE Computer Society Conference on Pattern Recognition and Image Processing, 1978, pp.220-226.

Gilloi, W., Interactive Computer Graphics, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1978.

Goldberg, M. P., Schmidt, A. H., and Chrisman, N. R., "Integration and Analysis of Multiple Geographic Data Bases: An Application of ODYSSEY." Harvard Library of Computer Graphics, 1979 Mapping Collection, Volume II, pp. 81-97.

Guptill, S. C., "A Digital Cartographic Data Base for Land Use and Land Cover and Associated Maps." Harvard Library of Computer Graphics 1979 Mapping Collection, Volume 2, 1979, pp. 99-106.

Hagan, P. J., "The Evaluation of a Network Data Structure for Cartographic Features." Proceedings of the ACSM, 1981.

Haralick, R. M., and Shapiro, L. G., "A Data Structure for a Spatial Information System." AUTO CARTO IV, Volume II, pp. 291-303.

Harary, F., Graph Theory. Addison-Wesley, Reading, Mass., 1971.

REFERENCES (Continued)

Honea, R. B., and Johnson, P., "Computer Mapping Software and Geographic Data Base Development: Oak Ridge National Laboratory User Experience." Harvard Library of Computer Graphics, 1979 Mapping Collection, Volume 2, pp. 119-130.

Horowitz, E., Shani, S., Fundamentals of Data Structures, Computer Science Press, Rockville, Maryland, 1978.

Hunter, G. M., and Steiglitz., "Operations on Images Using Quad Trees." IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume PAMI-1, 2, 1979, pp. 143-145.

IBID, pp. 201-250.

IBID, pp. 307-344.

_____, "Linear Transformation of Pictures Represented by Quad Trees." Computer Graphics and Image Processing 10 (1979), 289-296.

Ichikawa, Yajima, and Yamamura, "Retrieval of Image Features in Terms of Content-Addressing of Hierarchically Structured Image Data! *IEEE (1980).*

Jarvis, R. A., "A Computer Vision and Robotics Laboratory," Computer, IEEE Computer Society, June 1982, pp. 9-24.

Kawaguchi and Endo, "A Method of Binary Picture Representation and Its Application to Data Compression." IEEE TRANS on Pattern Analysis and Machine Intelligence, Volume PAMI-2, 1, 1980, 27-35.

Kirchoff, G., "Uber die Auflosung der Gleichungen auf welche man bei der Untersuchung der linearen Verteilung galvanischer Strome gefuhrt wird! Amer Phys Chem, (1847, 497-508.

Knapp, E. M., and Rider, D., "Automated Geographic Information Systems and Landsat Data: A Survey." Harvard Library of Computer Graphics, 1979 Mapping Collection, Volume 4.

Lawson, T. J., "Automatic Cartographic Feature Identification." Final Technical Report, RADC-TR-80-387, 1981.

REFERENCES (Continued)

Liles, W. C., and Nugent, E. D., Data Management Considerations for Large Cartographic Information Systems. Technical Papers of American Congress on Survey and Mapping, 1981.

Lin, S. B., and Chang, S. K., "GRAIN - A Pictorial Data Base Interface." Proceedings, IEEE Workshop Icture Data Description and Mangement, 1980.

Lougheed, R., McCubbney, D., and Sternberg, S., "Cytocomputers: Architectures for Parallel Image Processing," IEEE Workshop on Picture Data Description and Management, 1980, pp. 281-286.

Lucas and Gibson, "Introduction to GBT." Interactive Systems Corporation, 1981.

Males, R. M., and Gates, W. E., "ADAPT: A Digital Terrain Model-Based Geographic Information System." Harvard Library of Computer Graphics, 1980 Mapping Collection, Volume 8.

Mallgren and Shaw, "Graphical Transformation and Hierarchic Picture Structures." Computer Graphics and Image Processing 8, 1978, 237-258.

Martin, J., Computer Data Base Organization, 2nd ed, 1977, Prentice-Hall,Inc.

Marshall, C., Applied Graph Theory, Wiley and Sons, NY, 1971.

Mannos, J., "The Next Generation of Soft Copy Image Displays." Technical Papers of the American Society of Photogrammetry, 1980.

Milgram, D., "Constructing Trees for Region Description." Computer Graphics and Image Processing II, 1979, 88-99.

Milgram, D., "Note: Constructing Trees for Region Description," Computer Graphics and Image Processing 11, 88-99, 1979.

McEwen, R. B., "U.S. Geological Survey Digital Cartographic Data Acquisition." Harvard Library of Computer Graphics, 1979 Mapping Collection, Volume II, pp. 136-142.

Miller, S. W., "A Compact Raster Format for Handling Spatial Data." Technical Papers for the American Society of Photogrammetry Fall Technical Meeting, 1980, pp. CD-4-A-1, CD-4-A-18.

REFERENCES (Continued)

Morehouse, S., and Dutton, G., "Extraction of Polygonal Information from Gridded Data." AUTO CARTO IV, Volume II, 1979, pp. 320-327.

Narendra, P. M., and Goldberg, M., "Image Segmentation with Direct Trees." IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume PAMI-2, No. 2, 1980, pp. 185-191.

Nicholson, W. L., Major General. "Keynote Address." Photogrammetric Engineering and Remote Sensing, Volume 46, No. 7, July 1980, pp. 917-921.

Nyerges, T., "Representing Spatial Properties in Cartographic Data Bases." Technical Papers of the American Congress on Surveying and Mapping, pp. 29-41, 1980.

Pavlidis, T., Structural Pattern Recognition, Springer, Verlag, 1977.

_____, Algorithms for Computer Graphics and Image Processing, Computer Science Press, Rockville, Maryland, 1982.

_____. "Filling Algorithms for Raster Graphics." Computer Graphics and Image Processing 10, 8-15.

Perkins, W., "Area Segmentation of Images Using Edge Points." IEEE TRANS PAMI 2, 1 (Jan 1980), 8-15.

Peucker, T., "Data Structures for Digital Terrain Modules, Discussion and Comparison." First International Advanced Study Symposium on Topologic Data Structures for Geographic Information Systems, Volume 5, 1978.

Peuquet, D. J., "A Raster Mode Algorithm for Interactive Modification of Line Drawing Data." Computer Graphics and Image Processing 10, 1979.

_____. "The Need for Raster Processing of Cartographic Data." AUTO CARTO III, pp. 139-147.

Pfaltz, J. L., "Efficient Multi-Attribute Retrieval Over Very Large Geographic Data File." AUTO CARTO IV, Volume 1, 1979, pp. 54-62.

Pooch, U., Nieder, A., "A Survey of Indexing Techniques for Sparse Matrices." Computing Surveys, Volume 5, No. 2, (1973).

REFERENCES (Continued)

Pratt, W. K., Digital Image Processing, 1978.

Ranade, Rosenfeld, and Samet, "Shape Approximation Using Quad Trees." Pattern Recognition 15, 1 (1982), 33-41.

Reed, C., "Habitat Characterization: A Production Mode Usage of a Geographic Information System." Harvard Library of Computer Graphics, 1980 Mapping Collection, Volume 10, pp. 113-119.

Roach, R., Grover, D., and Lucas, D., "The Scan-Graphics Solution." Computer Graphics World, December 1980.

Rosenfeld and Samet, "Tree Structures for Region Representation." Intelligence Symposium on Computer Assisted Cartography (AUTO CARTO IV) (1979), 108-116.

Rosenfeld, A., "Extraction of Topological Information From Digital Images," First International Advanced Study Symposium on Topological Data Structures for Geographic Information Systems, Volume 6, Spatial Algorithms: Efficiency in Theory and Practices, Harvard University, 1970.

Rosenfeld, A., and Pflatz, J., "Sequential Operations in Digital Picture Processing, JACM, Vol. 13, No. 4, October 1966, pp. 471-494.

Samet, H., Connected Component Labeling Using Quad Trees, Computer Science Department, University of Maryland, TR-756, April 1979.

Samet, H., "Region Representation: Quad Trees for Boundary Codes." CACM 23, 3 (Mar 1980), 163-178.

_____. "An Algorithm for Converting Rasters to Quad Trees." IEEE TRANS ON PAMI 3, 1 (1981), 93-95.

_____. "Connected Component Labeling Using Quad Trees." (Apr 1979) "Computing Perimeters of Images Represented by Quad Trees." (Apr 1979) "Region Representation: Raster-to-Quad Tree Conversion (May 1979), Technical Report, Computer Science Department, University of Maryland, College Park, MD.

REFERENCES (Continued)

Schrock, B. L., "Applications of Digital Displays in Photo Interpretation and Digital Mapping." Technical Papers of the American Society of Photogrammetry, March 1980, pp. 201-212.

Shani, U., "Filling Regions in Binary Raster Images: A Graphic Theoretic Approach." ACM, 1980.

Shapiro, L. G., "Data Structures for Picture Processing: A Survey." Computer Graphics and Image Processing 11, 1979, pp. 162-184.

Shipiro, L. G., and Haralick, R. M., "A General Spatial Data Structure." Proceedings of IEEE Computer Society Conference on Pattern Recognition and Image Processing, 1978. pp. 238-249.

Sloan, K. R., and Tanimoto, S. L., "Progressive Refinement of Raster Images." IEEE Transactions on Computer, Volume C-28, No. 11, 1979, pp. 871-875.

Srihari, S., "Representation of Three-Dimensional Digital Images." Computing Surveys 13, 4, (Dec 1981), 399-424.

Tanaka, Tamura, and Tanaka, "Picture Assembly Using a Hierarchical Partial Matching Technique." IEEE Trans on Systems, Man and Cybernetics 8, 11, (1978) 812-819.

Tanimoto, S. L., "An Iconic/Symbolic Data Structuring Scheme." Pattern Recognition and Artificial Intelligence (C. H. Chen, Editor) Academic Press, New York 1977.

_____. "Image Transmission with Gross Information First." Computer Graphics and Image Processing 9 (1979), 72-76.

_____. "Hierarchical Picture Indexing and Description." IEEE (1980)

_____. "Pictorial Feature Distortion in a Pyramid." Computer Graphics and Image Processing 5, (1976), 333-352.

REFERENCES (Continued)

Tanimoto and Jackins, "Geometric Modeling with Oct Trees." IEEE (1980)

Vaidya, Prashant, D., Shapiro, L. G., Haralick, R. M., Minden, G. J., "An Experimental Relational Data Base System for Cartographic Application." Technical Papers of the American Society of Photogrammetry, March 1982.

Weller, D., and Palermo, F., Data Base Requirements for Graphics, IBM Research Report RJ2435 (32246), January 1979.

Wherry, D. B., and Friedman, S. Z., "Cartographic Applications of an Image Based Information System." AUTO CARTO III, Volume 1, pp. 148-168.

White, D., "ODYSSEY Design Structure." Harvard Library of Computer Graphics, 1979 Mapping Collection, Volume II, pp. 207-216.

Williams, O. W., "Outlook on Future Mapping Charting, and Geodesy Systems." Photogrammetric Engineering and Remote Sensing, Volume 46, No. 4, April 1980, pp. 487-490.

Williams, R., "Image Processing and Computer Graphics." Computer Graphics and Image Processing 10, 1979, pp. 183-193.

Youngmann, C. E., "A Linguistic Approach to Map Description," First International Advanced Study Symposium on Topological Data Structures for Geographic Information Systems Volume III, October 1977.

Zimmerman, E., "The Evolution of the Domestic Information Display Systems." AUTO CARTO IV, Volume 1, 1979, pp. 172-187.

Zobrist, A. L., "Data Structure and Algorithms for Raster Data Processing." AUTO CARTO IV, Volume 1, 1979, pp. 127-137.

APPENDIX A

ALGORITHM FOR PRODUCING A LABELED CONNECTED COMPONENT (FROM MILGRAM, 1979)

Algorithm for Producing the Labeled Region (Milgram, 1979)

```
"INITIALIZE"
  REGION_COUNT ← 1
  REGION_NOT_NEW ← FALSE
  NEW_LABEL_NEEDED ← FALSE
  PLACE BACKGROUND LABEL ON ACTIVE LIST
  DO  FOR X = 1,...,WIDTH "Label the outer boundary as background"
      CURRENT_LABELS(X) ← (REGION_COUNT,BACKGROUND)
                                                        OD

"PROCESS EACH NEW IMAGE RECORD"
 DO  FOR EACH IMAGE RECORD
     DO  FOR X = 1,...,WIDTH "Cycle the label buffers"
         PREVIOUS_LABELS(X) ← CURRENT_LABELS(X)    OD

      READ AN IMAGE RECORD AND CYCLE THE IMAGE ROW
       BUFFERS
      DO  FOR X = 1,...,WIDTH "Threshold and label each pixel"
          CALL DETERMINE_LABEL(X,LABEL,
           NEW_LABEL_NEEDED,REGION_NOT_NEW)
          CURRENT_LABELS(X) ← LABEL
          IF  NEW_LABEL_NEEDED
              THEN "We seem to be entering a new region"
                   PLACE LABEL ON ACTIVE LIST
                   NEW_LABEL_NEEDED ← FALSE    FI
          IF REGION_NOT_NEW
             THEN  "Merge the two equivalent region descriptions"
                   CALL EQUIV(PREVIOUS_LABELS(X),
                    CURRENT_LABELS(X - 1))
                   REGION_NOT_NEW ← FALSE      FI
          CALL DOSTATS(X) "Accumulate the feature data for X"
      OD
      DO  FOR EACH LABEL ON THE ACTIVE LIST BUT NOT IN
           CURRENT_LABELS
          "Delete any completed region from the active list"
          FIND LEAST X SO THAT PREVIOUS_LABELS(X) = LABEL
          PARENT_LABEL ← CURRENT_LABELS(X)
          DELETE LABEL FROM ACTIVE LIST.  PLACE ON
           COMPLETED LIST,
          APPEND LABEL TO THE CONTAINMENT LIST OF
           PARENT_LABEL
      OD
  OD
  "At this point the completed region list has the required tree
   structure"
  "Now compute the desired features from the accumulated feature
   data at each node"
  RETURN
  END
```

```
PROCEDURE DETERMINE_LABEL(POSN,LABEL,NEW,EQ)
"Assign a label to the lower right position of a 2 X 2 neighborhood"
COMPUTE C = VECTOR OF THRESHOLD DECISIONS FOR A 2 X 2
 NEIGHBORHOOD AT POSN,
CASE C OF
|00  01  11|
|00, 10, 11|
        DO  "Label the point with the upper-left label"
            LABEL ← PREVIOUS_LABELS(POSN-1)
  |00  10  11  01|
  |10, 10, 01, 01|
        DO  "Label the point with the upper-right label"
            LABEL ← PREVIOUS_LABELS(POSN)
  |01  11  10  00|
  |00, 00, 11, 11|
        DO  "Label the point with the lower-left label"
            LABEL ← CURRENT_LABELS(POSN-1)
  |10  01|
  |00, 11|
        DO  "Equivalence the lower-left and upper-right labels"
            LABEL ← PREVIOUS_LABELS(POSN)
            EQ ← TRUE
  |11|
  |10|
        DO  "Create a new background label"
            REGION_COUNT   REGION_COUNT + 1
            LABEL ← (REGION_COUNT,BACKGROUND)
            NEW ← TRUE
  |00|
  |01|
        DO  "Create a new foreground label"
            REGION_COUNT ← REGION_COUNT + 1
            LABEL ← (REGION_COUNT,FOREGROUND)
            NEW ← TRUE
  |10|
  |01|
```

```
        DO  "There is background equivalence and a new
             foreground label"
            REGION_COUNT ← REGION_COUNT + 1
            LABEL ← (REGION_COUNT,FOREGROUND)
            NEW ← TRUE
            EQ ← TRUE
ESAC
RETURN
END

PROCEDURE EQUIV(LABEL1, LABEL2)
"Two region labels identify the same region"

IF  LABEL1 ≠ LABEL2
    THEN "Merge the two region descriptions"
         COMBINE THE ACCUMULATED STATISTICS OF
          LABEL2 WITH LABEL1
         APPEND THE CONTAINMENT LIST OF LABEL2 to
          LABEL1
         DELETE LABEL2 FROM THE ACTIVE LIST
         DO  FOR X = 1,...,WIDTH
             IF  PREVIOUS_LABELS(X) = LABEL2
                 THEN PREVIOUS_LABELS(X) ← LABEL1   FI
             IF  CURRENT_LABELS(X) = LABEL2
                 THEN  CURRENT_LABELS(X) ← LABEL1   FI

         DO
FI
RETURN
END
```

APPENDIX B

DATA STRUCTURE EVALUATION OVERVIEW

APPENDIX B. DATA STRUCTURE EVALUATION OVERVIEW

Earlier work on the IRDS project has generated Technical notes describing basic issues on data structures (TEC-5) and data structure costing (TEC-14 and TEC 20). These ideas are combined to produce evaluation metric for data structures. The premises for the metric are as follows:

1. Multiple structures can support a given algorithm or set of algorithms.

2. A data structure is in support of the algorithm(s), rather than the algorithm is formed to use a predefined data structure.

Prior to applying the evaluation metric for a data structure, there is a necessary condition which must be met by the structure. That is, the structure must contain the data necessary to support the algorithm. There are two conditions which must be fulfilled to meet this criterion:

1. The data structure must be complete, (i.e., the structure must contain all the data needed for the algorithm). If the single structure cannot meet this criterion, it must be combined with the other data structures necessary to supply the complete set of data. The evaluation is then based on the aggregate of the data structures necessary to support the algorithm.

2. The data structure must be creatable from that source data available at the point in processing where the algorithm will execute for the first time. Not only must the data values exist, but the relations which form the structure must be known.

The general metric for evaluation of the data structure is:

VALUE = COST + QUALITY + EFFICIENCY + SUPPORT

where the individual components are normalized values. The terms are discussed below.

1.1 Value

The relative value of each of the factors must be based on the application requirements. It is obvious that the primary factor in a real-time system is efficency with cost perhaps the next most important factor. The driving factor on a rented computer time share system will probably be cost, whereas support and quality may be of greater importance on a dedicated computer system. The relative importance can then be accounted for assigning multipliers or percentages to each of the normalized component factors. For a real-time system one might use:

V = .2*COST + .05*QUALITY + .7*EFFICIENCY + .05*SUPPORT

where the weighting factors are arbitrarily assigned to match the relative importance of each category. For a dedicated developmental system the assigned weights might be:

V = .4*COST + .5*QUALITY + 0.*EFFICIENCY + .1*SUPPORT.

The relative importance is determined from the objectives and subjectively assigned weights for the final evaluation. Factors need not be included if they are of no real importance to the application, although it is hard to imagine a system where each of these factors is not of some importance.

1.2 Cost of a Data Structure

The COST portion of the evaluation metric refers to the space-time economics of data structures. COST is also the most difficult value to quantize. COST is the summation of the life-cycle costs associated with a particular data structure. The life cycle can be broken into three phases, dynamic creation, algorithmic data processing, and maintenance.

The costs are associated with algorithmic usage of a data structure. Often, multiple processes (algorithms) use the same data structure. This can be analyzed in two ways. The algorithms can be bundled and considered as a single complex algorithm or each algorithm or group of algorithms can be treated separately. If they are bundled, the complexity may make analysis an impossible task. If they are treated separately, many costs must be prorated (1) over each of the algorithms or treated as separate units (2).

(1) *TOTAL COST = CREATION COST + process-algorithm-1 +*
process-algorithm2 +...+ process-algorithm-n +
maintenance-1 + maintenance-2 ... etc.

(2) TOTAL COST = (.5) Creation-COST-1 + .5 creation-cost-2 + process-algorithm-1 + process-algorithm-2 + (.3) maintenance-1 + (.7) maintenance-2.

1.2.1 CREATION COST

Creation cost includes I/O costs from the storage media, the CPU costs for conversion of data to the data structure, memory usage costs of the data structure and any temporal files for creation of the data structure, and the algorithmic costs of data conversion. In other words, the creation of the data structure requires an algorithm(s) plus some I/O costs and memory usage costs prior to any usage of the data structure in support of the target process. These costs are implementation dependent and are calculated for the target machine configuration. These costs are calculated in the same manner as the processing costs in the next section, although the I/O costs are in general much higher. The costs are calculated over time T of the creation cycle.

CREATION COST = I/O + CONVERSION + STORAGE

1.2.2 PROCESSING COST

The processing costs are those directly associated with the algorithm's usage of the data structure. Determination of these costs requires careful analysis of the behavior of the algorithm. Simple algorithms may be linearly dependent on the volume of data, whereas complex algorithms may require the inclusion of alternative conditions. Page (PAGE, 1977) does an analysis based on 'ands' and 'ors' of various processes and probabilities of alternative processes to determine cost. To the processing costs must be added any I/O costs and memory usage costs over the time of the process.

1.2.3 MAINTENANCE COSTS

Maintenance costs are those costs over the life of data structures which are not incurred during the creation or algorithmic use of the data structure. The data structure may need to be updated before the algorithm is applied in subsequent passes, or it may be stored either in memory or on a storage media between algorithmic processes. If the structure is stored on a secondary media, the I/O costs must also be added.

MAINTENANCE COST = Update-cost + storage-cost + I/O cost.

1.3 Quality of a Data Structure

Unlike the COST portion of the metric which is based on economic values, subjective evaluation is added in the QUALITY factor. These are factors which are not sufficiently accounted for in COST, EFFICIENCY, and SUPPORT factors. Values for its various components and even the components themselves are determined from the application, environment, and future potential. The factors include such considerations as whether a data structure:

1. Supports a wider range of functions than essential.

2. Supports a more pleasing or more informative graphic display.

3. Is more easily understood by a wide range of people allowing shorter development times or use in other programs.

4. May be widely used for advanced research which promises future system growth, modernization, or longer system life.

These subjective ideas and others may play the most important role in the eventual selection of a data structure.

1.4 Efficiency of a Data Structure

Efficiency is the simplist component of the metric. It is the inverse of the time factor (T) for an average data retrieval, process, and store. It can be computed in any convenient unit such as microseconds or CPU cycles. The actual retrieval/store time may be near zero for simple or indexed structures or large where data searches are required.

If data transfers between main memory and secondary storage are required, these times should also be included in the effiency component.

EFFICIENCY = 1/T where T = t(retrieval) + t(process) +

t(store) + t(I/O)

1.5 Support for the Data Structure

The SUPPORT component of the metric is a subjective measure of the implementation difficulty on a hardware and software configuration for an algorithm and data structure. It is unusual for a system to be incapable of supporting an algorithm, but some cases do exist. For example, some of the high-level languages such as FORTRAN do not support recursion. The SUPPORT component is a combination of subjective evaluation of:

1. Programming Language Support - the relative ease of programming the algorithm in the available or chosen high-level language.

2. Software Support - the relative ease of the high-level language to create and use the data structure.

3. Primary and Secondary Storage Support - the amount of main memory and secondary storage available relative to the requirements imposed by the data structure.

4. Operating System Support - the capabilities of the operating system to handle the algorithm and data structure. The algorithm and data structure may require multitasking under one operating system and present no such problems under another operating system.

The SUPPORT component is:

SUPPORT = Programming ease + language-data structure compatibility + storage availability + Operating System capability.

SECTION 2. EVALUATION METRIC APPLIED TO FOUR CONTOUR FILLING ALGORITHMS AND DATA STRUCTURES

This example uses the evaluation metric for the contour filling algorithms; Ordered Edge list, Seed Fill Parity Check, and Complementary Fill described in IRDS Project Information Report TEC-37, "Comparison of Contour Filling Algorithms". The following basic assumptions are made:

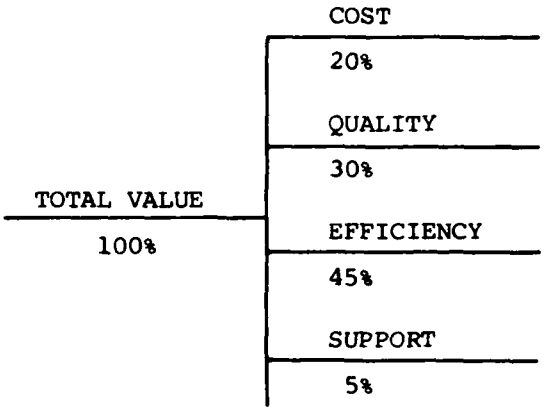
1. The computer is a standard third or fourth generation general purpose machine.

2. The display device is raster and has frame buffers.

3. Core memory is not a limiting factor.

4. Adequate disk storage is available.

The target system use is assumed as laboratory use to evaluate algorithms. The overall evaluation is based upon the target system goals where Efficiency is the prime concern, Quality is of second importance, Cost third, and Support is the lowest. The total is broken down as follows:



TOTAL VALUE = .2 Cost+ .3Quality + .45Efficiency + .05 Support

2.1 General Evaluation

After looking at the objectives, the next step is to look at the algorithms and characteristics as they apply to data structures. All of the algorithms use a similar primary data structure for raster display. Therefore, in this case our primary concern will be creation costs, I/O, and supplementary structures. Reviewing the characteristics of the algorithms we find:

1. Ordered Edge List:

   a. Has low I/O

   b. Can be done in memory or frame buffer

   c. Needs a presorted edge list

   d. Has been implemented

2. Seed Fill

   a. Has high I/O

   b. Needs seed pixel

   c. Needs Connected Components to prevent flooding or errors

   d. Can be used for parallel processing (not applicable to target configuration)

3. Parity Check

   a. Needs a presorted edge list

   b. Needs a line adjacency graph to prevent algorithm failure

   c. Has high I/O

4. Complementing Fill

   a. Can easily become I/O bound

   b. Slow execution

Our general assessment says that all the algorithms can be used but the additional supporting structures such as presorted edge lists and connected components must be created and maintained.

2.2 Creation Cost

The creation costs are high for presorted edge lists for the OEL and Parity, the connected components list for Seed fill and the Line Adjacency Graph for the Parity. Relative values are assigned as 10 for the least cost and 1 for the highest.

The processing and maintenance costs are assigned in the same manner. The requirement for the interior pixel in the Seed Fill increases the algorithm complexity.

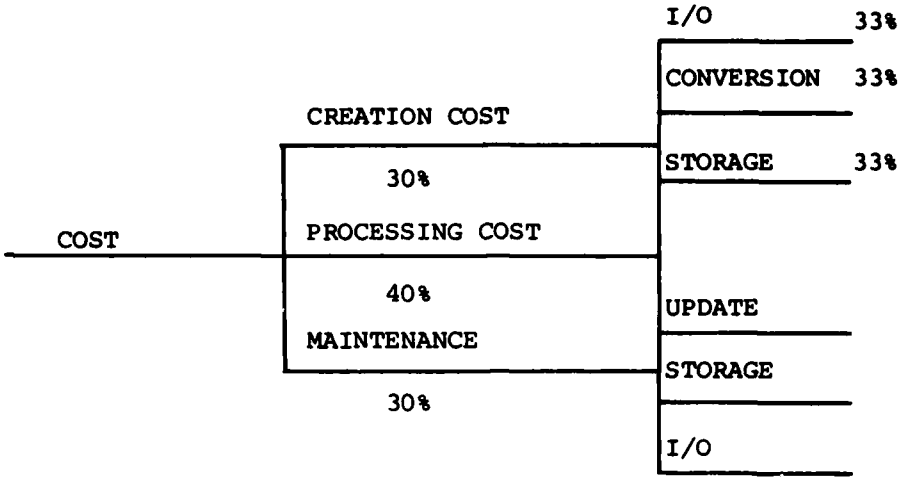
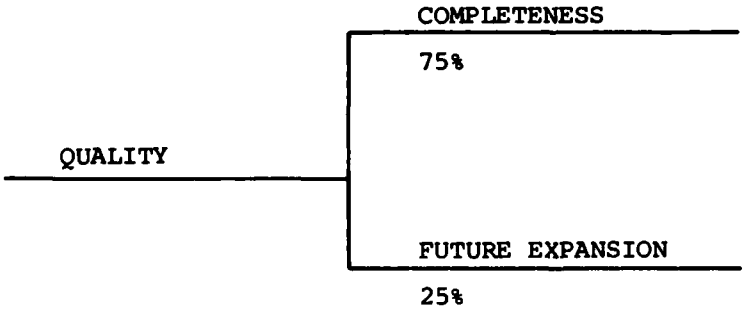


Table B-1 presents the values which are available for each of these costs for each filling algorithm.

2.3 Quality

The quality of a data structure is a subjective evaluation. In the case of these structures, the ordered edge list, connected components, and line adjacency graphs required by the structure are additional useful structures. The seed pixel for this appears not to be a useful addition. Our sample criteria says that Completeness will be 75% and Expandability will be 25%



A subjective evaluation is that the most complete structure is for the OEL because it has been implemented, has low I/O and has the ordered edge list. Second is the Parity Check, third the Seed Fill and last the Complementary Fill which has the least additional information. The following are assigning values we have:

Table B-1.

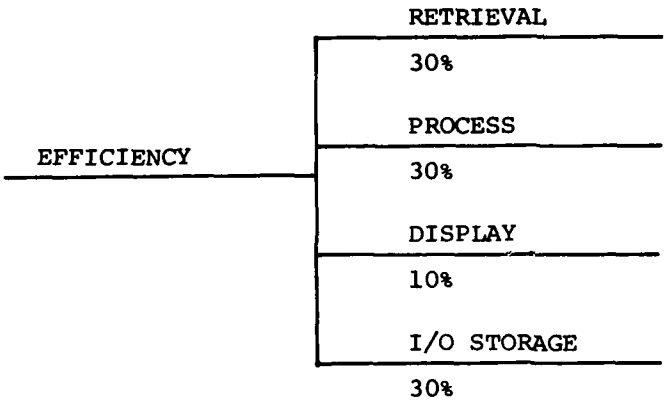
Data Structure Cost Values

	OEL	SEED FILL	PARITY CHECK	COMPLEMENTARY
CREATION				
I/O	5	5	1	10
Conversion	5	5	1	10
Memory	1	1	5	10
	11	11	7	30
÷ 3				
To Normalize	4	4	2	10
PROCESSING				
Complexity	10	2	6	1
MAINTENANCE				
Update	10	2	1	9
Storage	3	10	1	9
I/O	10	5	5	1
	33	17	7	11
÷ 3				
To Normalize	10	6	2	4
COST TOTAL	14	12	10	11
NORMALIZED TOTALS	10	8	6	7

	OEL	SEED FILL	PARITY	COMPLEMENTARY
COMPLETENESS (.75)	10	5	6	1
EXPANSION (.25)	8	10	10	1
NORMALIZED TOTAL	9	6	8	1

2.4 Efficiency

Efficiency measurements are taken from the comments on the data structure and our knowledge of the algorithms. The values are broken down as follows



In this case we do not have enough information to adequately evaluate Efficiency. The numbers given are masked by the combination of I/O times and execution times. Assigning some admittedly arbitrary values using the inverse of the expected time - so that the longest time is the lowest score we have:

	OEL	SEED FILL	PARITY CHECK	COMPLEMENTARY
QUALITY	6	8	10	1
RETRIEVAL.3	10	8	8	5
PROCESS.3	--	--	--	--
DISPLAY.1	--	--	--	--
I/O STORAGE.3				
TOTAL	36	36	38	26
UNVALUED USE 10				
NORMALIZE ÷ 4	9	9	10	7

2.5 Support

In this case we do not have support information and all four techniques will be valued equally (at 10) since all are targeted for the same system and will be implemented for that particular system.

2.6 Total Cost Evaluation

The final step is to total each of the subtotals and come up with a total score. (See Table B-2.)

The choice is not clearly decided since the totals are fairly close. The complementary fill is not recommended but any of the other data structures and algorithms may be a correct choice.

Table B-2.

Total Cost Evaluation

		COST		QUALITY		EFFICIENCY		SUPPORT
OEL	=	(.2)10	+	(.3)9	+	(.45)9	+	(.05)10
		2	+	2.7	+	4	+	½
SEED	=	(.2)8	+	(.3)6	+	(.45)9	+	(.05)10
		16	+	1.8	+	4	+	½
PARITY	=	(.2)6	+	(.3)8	+	(.45)10	+	(.05)10
		1.2	+	2.4	+	4.5	+	.5
COMPLEMENTARY	=	(.2)7	+	(.3)1	+	(.45)7	+	(.05)10
		1.4	+	.3	+	3	+	½

APPROXIMATE TOTAL

OEL = 9

SEED = 8

PARITY = 8½

COMPLEMENTARY = 5

APPENDIX C

RELEVANT DATA STORAGE TECHNOLOGY

APPENDIX C. RELEVANT DATA STORAGE TECHNOLOGY

This appendix is intended to present descriptions of the state-of-the-art in data storage and data base machine technologies which should be evaluated for a role in the hardware support environment. The material presented in this appendix is an extract from a final technical report entitled "High Capacity Dynamic Recce Technology, (RADC-TR-81-369), January 1982, by Frederick K. Frantz and Deborah J. Norris (pp. 5-27 - 5-83). The descriptions of data storage technology include; LSI/VLSI, magnetic tape storage, rotating magnetic memories, and optical disks. The descriptions of data base machines included in this appendix are of back-end processors and associative memory and differ from the original source document in that the descriptions of cellular logic devices, special purpose function architectures, information storage and retrieval, and multiprocessor systems were not included, although they are mentioned in the taxonomy of Section 1.2. Readers interested in technology assessment of those topics should reference the original document.

Of particular interest to the reader will be:

1. Section 1.1.1.1 - Semiconductor Random Access Memories which offer an important contribution in the memory planes of the interactive station.

2. Section 1.1.2 - Rotating Magnetic Disks which play an important role for both the host processor and interactive station. Current state-of-the-art offers 2.5 Gbyte capacity per disk, with a data transfer rate of 3 Mbyte/second and an average seek time of 16 MS.

3. Section 1.1.4 - Magnetic Tape Storage Techniques which discusses both mainframe computer compatible tapes (CCTs) and wideband tapes (WBT) which are necessary for archival storage.

4. Section 1.2.1 - Back-end Data Base Machines, particularly of interest should be the discussion of a comercially available relational data base machine, which offers a high degree of parallelism, high storage capacity, and random access addressing.

1.1 Data Storage Technology

The purpose of this section is to provide the reader with a brief overview of data storage technology. The scope of this review will be on hardware and generic types of storage devices. While the emphasis is on current capabilities, trends are noted along with the sources of these predictions.

The contents of this section include:

1. LSI/VLSI solid state storage technology, including:
    a. Bipolar and metal oxide semiconductor (MOS) circuits for random access memory (RAM).
    b. Magnetic bubble memories (MBMs).
    c. Charge coupled devices (CCDs).
    d. Gallium arsenide devices.
    e. Josephson junction devices.
    f. Electron beam addressed memory devices.
    g. Trends in VLSI circuits.
2. Rotating magnetic memories.
3. Optical disks.
4. Magnetic tape storage.

The contents of this section are not intended as a thorough or complete review of current or projected storage technology. They comprise a sampling of current reviews. Thus, the judgments and assertions repeated herein are those of the authors cited. Research to verify or substantiate the various assertions, assessments, and predictions was not considered to be part of this project.

This report is an exposition of factual information having some of the characteristics of an annotated bibliography, and simply presents findings related to the current state-of-the-art in storage technology and systems. The storage technology which will be described is relatively well-known and in a sufficiently advanced state of development to either be in wide use now or to have sufficient promise to anticipate its wide use. Two technologies, electron beam addressed memories and charge coupled devices, appear to

current reviewers to be receiving less support from vendors, due in part to the successes of competing technologies.

As mentioned above this brief survey of storage technologies and devices is not intended to be complete. We will not describe plastic film or plated wire or amorphous devices, for example. Nor is it intended to be thorough. We will not dwell at length on the differences between SOS, CMOS, NMOS, IIS, TTL, ECL, BCL, etc. Nor will we discuss the advantages of helical scan versus longitudinal. These are small in comparison with the differences, for example, between magnetic tape and solid state devices. Thus, the emphasis is on major differences between the differing technologies.

1.1.1 LSI/VLSI SOLD-STATE STORAGE TECHNOLOGY

In this subsection the capabilities of certain examples of the following types of LSI/VLSI solid-state storage technology are described:

1. Semiconductor random access memories (RAMs)
2. Magnetic bubble memories (MBMs)
3. Charge-coupled devices (CCDs)
4. Gallium arsendide devices
5. Josephson junction devices
6. Electron beam devices

The last part of this subsection summarizes the opinions about future technology trends of four executives of companies which are prominent in the field. They are: J. Fred Bucy of Texas Instruments, Ralph E. Gomory of IBM Corporation, Gordon Moore of Intel Corporation, and John A. Young of Hewlett-Packard Company.

1.1.1.1 Semiconductor Random Access Memories

A recent article in Electronic Design (March 31, 1981, p. 82), summarizing presentations at the Electro/81 Conference, described the impact of 64-Kbit RAMs as follows:

> "If reality lives up to expectations for semiconductor devices such as 64-Kbit dynamic RAMs and VLSI gate arrays, the practice of digital design will undergo radical changes before the end of the decade. Those who watch the semiconductor industry predict the 64-K chip. . . could become the most popular IC ever; usage may exceed 400-million units annually by 1986.."

At the same conference, Fred Jones of Mostek Corporation raised several challenges which the move from 16-Kbit to 64-Kbit RAMs has produced. These included:

1. The smaller size of each storage cell, making signal-sensing more difficult.

2. Smaller geometries, (e.g., 2.5 to 3.0 mm).

3. Single-supply operation.

4. New architecture schemes.

5. Increased operating margins that ensure manufacturability and reliability.

6. Uncertainty of standard for refresh cycle period value.

Assuming that the 64-Kbit design and manufacturing difficulties can be overcome, Daniel Kleskin of Dataquest predicted that the chip may have a worldwide market in excess of $4 billion by 1984. However, because of the long qualification cycles involved, he does not expect 64-Kbit RAMs to appear in computer systems until 1982. Between 1982 and 1986, the unit cost is expected to drop dramatically from $10 to $2 per chip, according to the Electronic Design article. At four times hardware, chip and assembly cost (assuming 120 memory chips and 24 logic chips plus equal board hardware and assembly cost) a Megabyte memory board could cost about $2500 in 1986. This is about one fifth of the current cost for a 1 Mbyte memory board for a VAX 11/780.

The remainder of this subsection is based largely on two articles by Eugene R. Hnatek (Hnatek, 80a) and Dave Bursky (Bursky, 80).

In the first article, Mr. Hnatek discusses bipolar RAMs, including storage circuits based on transistor-transistor logic (TTL) and emitter coupled logic (ECL) . In addition he describes various kinds of metal oxide semiconductor (MOS) devices, including both dynamic and static RAMS. Hnatek's article appeared in January 1980, but was current only to 1979. Nevertheless his near-term predictions apply to the time of this writing, and will be interpreted as current. In this part of the subsection, we will summarize Hnatek's discussion of progress in the making for the various types of RAMs, such as bipolar or MOS.

Bipolar transistor-transistor logic (TTL) RAMs using recent oxide-isolation technology in combination with various processing techniques can provide high speed and high density stores for cache, scratch-pad, and buffer memory applications. As the speed of microprocessors and other central processing units increase, the use of slower MOS memory is becoming

unsatisfactory. Thus higher speed RAM is becoming mandatory for main memory as well. Representative of the trend away from transistorized flip-flops is the Fairchild isoplanar RAM. It has small die size, higher yield, and lower unit cost than even the comparable dynamic n-channel MOS (NMOS) memories, Bipolar TTL RAMs have almost reached their speed plateau of about 20 to 25 nanoseconds access time. The current developments are in the 16-Kbit size. It remains to be seen if a successful bipolar TTL RAM with this size and speed can be produced at yields and manufacturing costs which are competitive with other technologies, such as static NMOS.

Emitter-coupled logic (ECL) is a circuit design technique which minimizes the effects of various stray capacitances and delays which are inherent in bipolar TTL circuits. With the emergence of ECL gate arrays and bit-slice citcuits, ECL system speed became impressive. ECL RAMs continue to be used in speed-intensive cache, buffer memory, and stores for constants in conjunction with such fast processors. Currently there is increased activity in ECL circuitry. Ten nanosecond access time is typical for a 1-Kbit device. By 1982 to 1983, Hnatek predicts this will be reduced to from 2 to 5 ns, and for a 4-Kbit RAM, 10 to 15 ns. The future of bipolar RAMs appears to lie with ECL, particularly in forthcoming super-computers.

There are two major categories of MOS RAMs - dynamic and static. Dynamic RAMs require refreshing at periodic intervals, (e.g., once every few milliseconds) in order to preserve their data. Static RAMs are bi-stable regenerative and do not require this refresh overhead. Previously, dynamic MOS RAMs have been refreshed by the action of control and timing circuitry which is external to the memory chip. Now a hybrid variety of MOS RAM, called pseudostatic MOS RAM, is being developed which incorporates a semi-automatic refresh feature within the chip's architecture.

Both static and dynamic MOS RAMs have access times which overlap those of some bipolar RAMs. In addition they feature lower power drain, and are well suited to applications in which power is a premium. New high speed static RAMs are being used in traditional bipolar RAM applications such as cache and writable control stores.

In static MOS RAM cells a bit is stored in a bi-stable flip-flop, as in bipolar RAMs. In dynamic MOS RAM cells, a bit is stored as an electrical charge in a capacitor of a single transistor cell. Leakage of this charge is the cause for the refresh requirement.

Static MOS RAMs tend to be faster than their dynamic counterparts. Also they are designed to protect against false or ambiguous operation and are less subject to noise generated by electrical current surges. They are most suited to critical control subsystems and/or higher electromagnetic interference environments. They require less interface and support circuits.

However, dynamic memories are denser and usually cost less. They are simpler and use less silicon. The trend is towards merging certain characteristics of static and dynamic designs. The newer chips of both types operate from a single 5 volt power supply, require low stand-by power and to some extent are becoming byte organized.

There have been significant changes in memory storage cells over the last several years and the trend is continuing with the use of new mask lithography techniques, etc. Density changes are most striking. Bipolar memory cell sizes have gone from an average 30 mil^2 for a six to eight transistor Schottky TTL circuit to 0.68 mil^2 for a merged two-transistor cell. MOS cell sizes have gone from 2.15 mil^2 for a PMOS three transistor circuit to 0.3 mil^2 for an NMOS single transistor triple-layer polysilicon circuit. Table 1-1 shows the memory cell size evolution resulting from new developments. National Semiconductor is using NMOS one transistor triple poly storage cell in its 64-Kbit RAM. This approach not only reduces cell size, but also provides a higher degree of immunity to soft errors caused by ionizing particles. Texas Instruments is using a taper isolated cell which eliminates the need for a separate storage capacitor for its dynamic RAMs under development. This approach is likely to appear, according to Hnatek, in a 256-Kbit dynamic RAM.

In large capacity memories, the dissipation of power per bit must be kept low in order to keep power supply and cooling requirements within bounds. It is essential that the memory element dissipate very little power when not being accessed. For single CPU operation, only one memory cell is being accessed at a time. This leaves nearly the entire memory in a quiescent state. Dynamic RAM is well suited to these conditions because the ratio of active to standby power (for most 4-Kbit RAMs) is about 20 to 1. Fortunately, the reduction in cell size by new developments coupled with scaling in size for a given technique, as well as the application of techniques which inherently use less power, have resulted in a significant reduction in power per bit. For example, as shown in Figure 1-1, the power per bit for a 4-Kbit RAM (in a 256-Kbyte 16-bit per word memory) averaged about 10 microwatts (for one word active at all times and 2 millisecond refresh rate). For a 64-Kbit RAM (in a 4-megabyte 16-bit-per-word memory), the average power dissipation per bit is about 0.6 microwatts.

Another trend has occurred, which has aided the solid state device manufacturers and users as a group. It is the adoption of industry standards for packaging and interconnections. A variety of approaches were generated for 4-Kbit RAMs. Manufacturers of 16-Kbit parts have adapted Mostek's 16-pin dual-inline package (DIP), thus eliminating a major source of confusion. There still are certain variables concerning 64-Kbit RAMs about which no standards have been developed, such as refresh rate and what to do with pin 1, but these are surmountable minor problems.

The 64-Kbit RAM is currently considered not only a major milestone in data storage, but also a bridge between large scale integration (LSI) and very large scale integration (VLSI). Hnatek described it as the precursor of increased computer performance in a smaller physical volume, at significantly lower cost. It represents the start of a new cycle of memory product design. It has resulted from combinations of fabrication developments such as projection versus contact printing, positive rather than negative photo resists, electron beam fabricated masks and dry etch. He predicted it will decline in cost faster than previous generation devices. Fewer and simpler printed circuit boards and reduction in power requirements and the number of supplies will allow total system costs to decline even faster.

The 256-Kbit dynamic RAM will require the new production techniques associated with VLSI circuitry, as well as new or modified fabrication processes and circuit design innovations. Hnatek predicted the availability of the 256-Kbit RAM sometime in 1983 or 1984.

Significant improvements have been appearing in static NMOS and complimentary MOS (CMOS) static RAMs as well. Various power-down strategies such as power gating, ready signal edge activation have made it possible for static RAMs to challenge dynamic RAMs in the area of power dissipation. Scaling techniques have provided high speed static RAMs, thus challenging the domain of bipolar RAMs. A great variety of static RAMs are now available, even in the 64-Kbit size, many of which are byte oriented. Nine bit versions are used for parity error detection.

NMOS and CMOS represent the major trends in static RAM fabrication processes with CMOS Now approach NMOS in terms of speed, density, and cost. LSI CMOS circuits are expected to benefit from increased development and production effort. One development now considered key to CMOS development is silicon-on-sapphire (SOS) technology. SOS fabrication has improved packing density due to the elimination of guard bands used in standard CMOS technology. The number of fabrication steps are thus reduced, resulting in lower costs and higher yields. If production and sapphire wafer costs can be reduced as a current effort of Hewlett-Packard is attempting, CMOS/SOS will be even more competititve.

Bursky asserted in his article that nearly a dozen companies will have 64-Kbit dynamic RAMs by the end of 1981. He notes that even though 64-Kbit dyanmic RAMs are basically pin compatible, circuits differ greatly. Bursky sees three "controversial" issues which have to be resolved:

1. What features to offer on the undefined pin 1, which becomes available as a result of switching from three power supplies to one.

2. How to handle refresh timing.

3. How to protect the memory from stray alpha particles.

Memories where pin 1 is left undefined are aimed at manufacturers of large systems in which a separate subsystem will handle refresh timing and/or at those who want the fewest possible operating modes in order to simplify testing. The majority of manufacturers have designed the no-function type.

A few manufacturers, Motorola, Mostek, and Inmos, have designed auto-refresh techniques for their 64-Kbit dynamic RAM chips. Two employ pin 1 with the option of operating in the no-function mode if pin 1 is not used. Motorola's chip offers two on-chip refresh operating modes, automatic and self-refresh. Mostek includes some built-in test capability. Inmos has taken the middle of the road, according to Bursky, by not assigning a function to pin 1 but making use of other timing signals to initiate an automatic refresh cycle.

Currently 16-Kbit chips are still more cost effective than are 64-Kbit chips. Figure 1-2 gives cost/bit curves which are attributed to Intel's projections. Most manufacturers have extended the economic crossover point well into 1983. The reasons, according to Bursky, are the currently low prices of 16-Kbit chips, as well as the availability of single supply 16-Kbit chips which are smaller and faster and which have more devices per wafer than previous three supply versions.

The characteristics of semiconductor RAM memories are characterized in Table 1-2. Since the various technologies have different characteristics, representative values have been included, mostly based on the articles cited in the text. It should be noted that advances in this area are being made at a rapid rate. As indicated in the above paragraphs, there are likely to be significant changes in these characteristics in the near future.

1.1.1.2 Magnetic Bubble Memories

In this subsection the current magnetic bubble memory (MBM) storage technology is described and compared to semiconductor RAM and magnetic disk, its closest competitors. Here again, the material used herein is based primarily on two articles (Hnatek, 80b) and (Swanson, 80).

A typical MBM module contains one or two 256-Kbit or 1-Mbit memory (Rockwell has demonstrated a 4-Mbit device developed for a military application. It could be in production by 1983 if Rockwell decides to continue to support MBMs.) packages, or chips, and various support circuits such as controller, function timing generator, data corrector/formatter, coil drivers, read and write function drivers and sense amplifier(s). The

controller handles single or multi-block transfers of data to the host at a rate of up to 200 Kbit/sec.

Swanson characterizes bubble memories as having small size, requiring no standby power, being non-volatile, having non-destructive readout and moderate access time. Furthermore, the high storage density and batch-production processing requiring relatively few masks and no diffusions offer the promise of lower cost per bit than other solid-state memory devices. Hnatek asserted that if a storage density greater than 64-Kbits per chip is needed, the logical choice would be magnetic bubble technology.

Magnetic bubbles are small cylindrical magnetic domains which reside in a thin synthetic ferrite or garnet film which has been epitaxially grown on a non-magnetic substrate. The domain and the garnet material have opposite magnetization. The "bubbles" are formed in the presence of an external magnetic field applied perpendicularly to the plane of the film using permanent magnets. A rotatable magnetic field, induced by quadrature currents in orthogonally wound coils, is used to move the bubbles along a path outlined by a deposited layer of aluminum or copper coated by a layer of permalloy. The presence or absence of a domain at a specific location defines a zero or one bit value. The bit streams move in a loop much like a serial shift register. Although each loop operates with serial input or output, the MBM device can be structured to transfer bits in parallel.

George Neeno states that MBM devices are organized in long serial loops or broken up into several minor loops which interface a single common loop called the major loop. Each configuration is claimed to have advantages suiting it to specific classes of applications. The serial MBM is described as much simpler than the minor/major loop version, but requires much longer average access time. Applications that do not need the faster access time, like memories for storing program steps sequentially, storing text strings, or logging large amounts of data, can taken advantage of the much simpler interfacing requirements of a serial loop (Neeno, 80).

MBM architectures can be used for associative searching in relational data bases. In addition to the use of the major/minor loop MBM device, an external marker memory and processing unit provides the associative capability. Each bit in the marker memory stores the results of a query applied to its corresponding page. A systematic series of access strategies and architectural modifications are described which are intended to improve potential system performance (Doty, Su, and Greenblatt, 80).

MBMs can be viewed as a solid state implementation analogous to rotating electromechnical memories such as disks or drum. However, the access time is faster, about five milliseconds for a 256-Kbit unit. Large disk systems derive their low cost per bit by spreading the high fixed electromechanical costs over a very large number of bits. Bubble memory components, however,

permit the memory system to be expanded in smaller increments with the cost per bit staying more constant due to smaller fixed costs. The two technologies can be combined into a hybrid memory system when fast repeated access is required for only a portion of the stored data for any one period (Hnatek, 80b).

Current costs of about one dollar per thousand bits for MBM devices are expected to drop as the production technology matures and volume increases. Currently, MBM is cost competitive in harsh environments, (e.g., where severe vibration, and/or a contaminated atmosphere exists, and/or continual access is required). These situations could occur for a field-deployable version of ASE.

How rapidly bubble memories achieve equality with silicon semiconductor and disk memories depends, according to Hnatek, on the ability to solve the following problems (Hantek, 80b).

1. The significant difference of MBM and RAM operation, resulting in diverse design rules.

2. The lack of standarization in package, input, organization, function, and architectures.

3. The need for test and characterization equipment, as well as support circuits, to become generally available.

4. The general lack of second sources.

Hnatek states that because of these problems, bubble chips will not be in high volume production in the near future. He also notes that new architectures and path geometries have been demonstrated which are capable of obtaining 25Mbits per square inch media storage density. Furthermore, he states that garnet chips have fewer defect densities than silicon, use smaller cells, and use simpler processes. Five to ten megabit MBMs are expected by 1985. What will become available, of course, will depend on the corporate commitment to continue MBM development and solve the above problems. The high momentum of semi-conductor technology may indicate that MBM will not be an economically viable alternative.

Although the serial bit shift is orders of magnitude slower than the fastest semiconductor shift register, the use of parallel and associative modes of operation may achieve high system performance. MBM's also have some advantages over charge-coupled devices (CCDs) in terms of non-volatility and packing density, but suffer from slower access times and data transfer rates. These characteristics are summarized in Table 1-3.

Table 1-3.

Magnetic Bubble Memory Characteristics

CHARACTERISTICS	VALUE	COMMENT
Storage Capacity	1 Mbit Module	
Access Time And Transfer Rate	Access Time = 4 to 11 milli-seconds (256 Kbit module) Transfers up to 200 Kbits per second	
Cost	$1 Per Thousand Bits	Current corporate commitment must be increased, as well as standardization, before MBM will be produced in volume.
Technological Risk	Medium	
Environmental Requirements	Medium	Power dissipation is two to three times higher than semiconductor RAM.
Integrity And Robustness	High	Especially effective in hostile environment.

1.1.1.3 Charge Coupled Devices

Charge coupled devices (CCDs) are, according to (Hnatek, 80b) low cost alternatives for bulk storage applications, filling the void between magnetic memories and semiconductor RAMs. Charge coupling is the process by which mobile minority charge carriers are collectively transferred from one semiconductor storage element to a similar, adjacent element, by manipulating the external voltages. Information in each element is represented by the amount of electrical charge present. Because the storage elements are interconnected through the substrate, space requirements are reduced. CCDs are manufactured on the same production facilities as are MOS RAMS, and have benefited from such fabrication techniques.

Three types of architectures are used for CCDs. These are:

1. serpentine (synchronous)

2. line addressable RAM (LARAM)

3. serial-parallel-serial (SPS)

Each architecture has its own cost/performance tradeoffs. For instance, serpentine is the simplest organization. This architecture has a wide operating frequency range, good density, average latency, but high power and clock loading. LARAM is a hybrid of CCD and RAM architectures. While this type has low power dissipation and clock capacitance, and excellent latency, it lacks the high density required for low cost, and has a limited frequency range. SPS offers the best organization for high density, and requires little overhead logic. Its lower power dissipation is offset by a limited frequency range and relatively poor latency (Hnatek, 80b).

The advent of the 64 Kbit dynamic RAMs with their higher density and lower cost, coupled with CCD processing and design problems which resulted in lack of product availability, has led several companies to place their CCD operation in a suspended state.

A few companies have announced experimental 256 Kbit CCDs. However, with magnetic bubble memories providing higher densities and lower costs per bit, uncertainty of the manufacturers; plans have combined to cast a shadow of doubt on the future of CCDs. However, for comparison purposes, we include as Table 1-4 a summary of CCD characteristics.

1.1.1.4 Gallium Arsenide Devices

The topic of this subsection is gallium arsenide solid state semiconductor integrated circuit technology which has been recently gaining momentum. Gallium arsenide devices (GADs), e.g., diodes, have been used for many years in microwave circuits operating above a gigahertz. Currently,

Table 1-4.

Charge Coupled Device Characteristics

CHARACTERISTICS	VALUE	COMMENT
Storage Capacity	64 Kbit Modules	
Access Time And Transfer Rate	Access Time = 400 microseconds Transfer Rate = 5 Mbits per second	Comparable or slightly faster access time than magnetic bubble memory, but orders of magnitude slower than sémiconductor RAM memory.
Cost	High	Relative to competing technology
Technological Risk	High	Current status of CCD development in several companies is suspended; currently the technology does not appear competitive with magnetic bubble memory or semiconductor RAM.
Environmental Requirements	Medium	Depends on the architecture
Integrity And Robustness	Medium	More susceptible to errors than magnetic bubble memory

some aerospace and computer companies are developing medium scale integrated logic circuits using GAD. It is claimed by Hnatek (quoting [F. A. Blum, 79]) that gallium arsenide is of great interest for its VLSI potential.

Gallium arsenide integrated circuitry has the potential for gate switching times of 100 picoseconds. Although one micron lithography contributes to this speed, the material has an inherently low power dissipation propagation delay product. It is the low dynamic switching energy of less than 0.1 picojoule that suggests its potential for high density devices. The low power, high speed advantage of gallium arsenide metal semiconductor field-effect transistors (MESFETs) integrated circuits stems directly from the high electron mobility and semi-insulating substrate which gives it high transconductance and unity gain bandwidths of approximately 80 gigahertz for a one micron gate device (compared with about 12GHx for a similar silicon MESFET).

There are several directions which GAD technology is heading. The enhancement device operating mode, using direct coupled FET logic (DCFL) comprises MESFET, JFET, HJFET, NAD MESFET devices. The depletion mode includes buffered FET logic (BFL) with MESFET active devices, or Schottky diode-FET logic (SDFL) with MESFET FAST Diode active devices. The latter approach appears to hold much promise for LSI circuit applications. SDFL devices are achieving comparable propagation delays (less than 75 picoseconds) at much lower power levels than BFL.

GAD MESFET LSI circuits exhibit a sixfold speed advantage over silicon MESFET circuits having the same power delay product. GADs have 25 to 40 times lower power dissipation than silicon for the same gate delay. Finally, GADs will operate up to 200° C without significant changes in characteristics. Even though the potential exists for exceptional performance, the skeptics point to the necessity to move GAD technology from the laboratory to the production line. Hnatek claims it will take another five years to know the potential of gallium arsenide integrate circuits (Hnatek, 80b). Estimates of GAD characteristics can be found in Table 1-5.

1.1.1.5 Josephson Junction Circuits

Josephson junction circuitry is a solid state integrated circuit technology which, like gallium arsenide devices, promises very high speed operation. Although the proposed fabrication of such devices resembles that of semiconductor solid state, the underlying physical principle which allows state switching is quite different. These circuits are based on four physical properties, as expressed by Hnatek (from [Anaker, 79]):

1. Certain materials, known as superconductors, lose all resistance to flow of electric current below a certain threshold temperature, usually near absolute zero.

Table 1-5.

Gallium Arsenide Device Characteristics

CHARACTERISTICS	VALUE	COMMENT
Storage Capacity	High	Potential for high density is based on low switching energy; potential is for modules up to a gigabit in size.
Access Time And Transfer Rate	Low	Potential exists for very high speed gate switching times; GAD LSI has exhibited an increase by a factor of six other similar silicon circuits.
Cost	High	
Technological Risk	High	Much more research is required to bring GADs into production
Environmental Requirements	Low	Excellent power dissipation and heat resistant.
Integrity And Robustness	Unknown	

2. Magnetic flux can be trapped in superconducting rings when linked to such circulating currents.

3. Electron pairing which occurs when a material becomes superconducting leaves a superconducting energy gap similar to that of semiconductors.

4. A sandwich of two superconductors separated by a very thin oxide layer allows a current to flow by a tunneling mechanism.

Researchers in several corporations have developed Josephson junction devices for the four basic logic elements felt necessary for total digital computer technology: OR, AND, INVERT, and LATCH. IBM staff have developed what they call current injection logic which exhibits a 30 picosecond gate delay and average dissipation of several microwatts per gate (Hnatek, 80b).

In contrast with voltage driven semiconductor logic, Josephson junction circuits are current driven. This duality between the roles of voltage and current leads to circuit differences such as fan out being achieved by connecting the several inputs in series rather than in parallel.

Storage of binary data in memory cells relies on the phenomenon of trapping magnetic flux in a superconducting ring, in association with the flow of current in a superconducting ring. As long as the current remains constant, the flux will remain. A Josephson junction in the loop is used to interrupt this flow, and to change the state of the memory cell.

Under development are two types of Josephson junction RAM cells: a nondestructive readout (NDRO) cell for use in ultrafast access arrays, and a memory cell for high density arrays designed to operate in the destructive readout mode. Memory cells, drivers, and decoders needed for the construction of the fast RAM have been experimentally evaluated. The results suggest that a NDRO array of 2-Kbits could have a one nanosecond access time, dissipate about 5 milliwatts and fit on a 6x6 mm chip. Semiconductor devices providing the same function would currently require about 200 times the access time and about 1000 times more power.

Commercially available Josephson junction devices are expected to be available in 1985. However, it should be noted that a good portion of the super computer which uses these super fast devices will be superconducting and have to exist inside a cryogenic bottle at 4° Kelvin. Table 1-6 provides a summary of Josephson junction characteristics.

Table 1-6.

Josephson Junction Circuit Characteristics

CHARACTERISTICS	VALUE	COMMENT
Storage Capacity	2 Kbit Array	
Access Time And Transfer Rate	Access Time = 1 nanosecond	
Cost	High	Significant development costs must be incurred before Josephson junction circuits will be available.
Technological Risk	High	Only experimental research has been conducted.
Environmental Requirements	High	Power consumption is low, but a large part of the system must be kept at 4° Kelvin.
Integrity And Robustness	Unknown	

1.1.1.6 Electron Beam Memory Devices

This subsection is based on an appraisal of electron beam memory technology contained in by a November 1977 technical report to NASA Goddard Space Flight Center, prepared by RCA/Government Communications Systems, under contract NAS5-24170, and authored by R. V. Kadyszewski (Kadyszewski, 77).

Electron beam memories are characterized by Kadyszewski as being capable of both high speed and high density. The information is stored and read from a reversible action type target which is typically made out of silicon dioxide. A high resolution electron beam is used to store and retrieve data stored as electrostatic charges associated with specific locations on the target film. This all takes place inside a vacuum.

Storage tubes with capacities ranging from 128 Kbits to 32 Mbits have been developed. Access time to a block of data stored in electron beam memory systems, consisting of an array of such tubes and support circuitry, ranges from five to thirty microseconds. Once the block has been found, the data can be read at about a 10 Mbps rate.

Kadyszewski forecast an increase to 100 Mbit per tube storage capacity and a 100 Mbit per second transfer rate, and operational systems by 1985. The cost per Kbit was stated as $0.20. micro-bit's System 7000 and General Electrics BEAMOS system were representative of the status of electron beam technology as of 1977.

Electron beam technology is not as suited to archival storage as are other technologies. First, the charge pattern will last for only a few months. Furthermore, a memory requiring 5×10^{14} bits would require about 10^9 watts of power, 10^7 cubic feet of space and ten billion dollars! This storage technology is more suited to smaller size systems which can utilize the access speed available, and tolerate the characteristic storage decay times. Regardless of the application, electron beam storage devices represent a high risk development effort. No operational and environmental data had been compiled for these systems (Kadyszewski, 77). Other data that has been developed for electron beam technology is included in Table 1-7.

1.1.1.7 Technology Forecast for Semiconductor Solid State Devices

We conclude this discussion of semiconductor solid state storage devices by presenting the views of four prominent corporate leaders in solid-state device technology. They are (alphabetically):

1. O. Fred Bucy (Texas Instruments, Inc.)

2. Ralph E. Gomory (IMB Corporation).

Table 1-7.

Electron Beam Memory Characteristics

CHARACTERISTICS	VALUE	COMMENT
Storage Capacity	32 Mbit Tubes	
Access Time And Transfer Rate	Access Time = 5 to 30 micro-seconds	
Cost	High	Development will require some support; eventual production cost should be competitive
Technological Risk	High	Operational systems are not anticipated before 1985.
Environmental Requirements	Unknown	
Integrity And Robustness	Unknown	

3. Gordon Moor (Intel Corporation).

4. John A. Young (Hewlett-Packard, Co.).

The prophecies summarized herein appeared in the January 8, 1981 issue of Electronic Design. This article also predicted for the next decade such advances as:

1. A 50-Mbyte floppy disk by 1990.

2. Future magnetic recording densities from 30 to 50 Mbits/in^2 (near the theoretical maximum).

3. $2.5x10^{14}$ bytes storage by 1983 using perpendicular recording on thin film metal particle media.

4. More use of cache buffers, intelligent disk drives, intelligence controllers and back-end data base processors will occur during 1981.

All four leaders pointed to advances in semiconductor process technology and very large scale integration (VLSI) as the major advances in storage technology during the 1980's. The major motivation will be the continuing requirement for greater memory capacity and higher storage density. Each provides a different perspective on additional developments that can be anticipated in the next decade. Other common themes to be noted include increasing reliance on computer aided design (CAD) of hardware, development of new production techniques, and the rapid advances anticipated in all of the areas mentioned in this section.

According to Bucy, 64-Kbit dynamic RAMs are in production today, and we can look forward with certainty to a 256-Kbit dymamic RAM in the next few years. Byte-wide memories in the form of wide-word static RAMs and EPRCMs are also becoming important, to serve distributed-computing and microprocessor systems. Microprocessors will evolve during this decade, approaching the performance of today's large mainframe computers, addressing very large memories, and having exceptionally high computational rates. Design utility systems will let the equipment designer model, simulate, verify, and generate test patterns for his/her specific program, then transmit the design data to the semiconductor manufacturer. Electron-beam lithograph technology will become a principal tool for the manufacture of VLSI components.

Lastly, the cooperative efforts of the physicists, circuit designers, and dedicated manufacturing personnel, coupled with automated manufacturing equipment and fail-safe redundancy in the design, will let technology evolve to a point where the failure of a semiconductor device will be an unusual event.

Ralph Gomory provides the following observations. Different techniques of lithography will be applied according to the following tradeoffs between production costs and speed and density: full wafer light-projection systems are the least expensive, but to obtain the resolution necessary for 1 micron resists is unlikely. Step and repeat systems have a better chance for 1 micron lithography but are more complicated, and more expensive to use. Electron-beam or X-ray systems have the necessary degree of precision, but given their elaborateness and slowness, they must not be economical.

Gomory also predicted that to achieve a one nanosecond cycle machine, the CPU must be restricted in size. This means better heat extraction and/or lower heat production techniques must be found. Josephson junction technology seems to fit in here. To use it however, a complete function must be done very fast. Gallium-arsenide logic has speed potential, though not as much as Josephson junctions. However, it can work at room temperature. Another key challenge is how to split up the functions among multiprocessor systems. Fiber optics may provide higher-bandwidth longer-distance links between CPUs and disk controllers. Multiplexing such connections remains a design problem and may remain so until it is clearer what data transfers will be made. Bubble memories are the leading contender for future large data base systems which have a hierarchy of storage devices. Optical disks are likely to provide much higher storage densities than can magnetic media.

According to Gordon Moore, as microelectronic technology produces smaller, higher density devices, no major tradeoffs are needed. Devices get faster, consume less power, and even get less expensive, as long as yield can be maintained. Scaling line widths and thicknesses appropriately will require new technology. The introduction of electron beam and X-ray lithography will not in itself be sufficient. If we could make half-micron lines we wouldn't know what to do with them. Also, tunnelling effects and voltage reductions pose new problems. The industry is waiting for good commercial machines, for X-ray resists which work in production and the ability to make low-defect masks required for X-rays. A micron feature size requires a quarter micron alignment tolerance or better. Intel has achieved this with simple Gallium-arsenide and bubble-memory structures. The minimum practical feature size for more complex structures is about 2 microns. Automatic local alignment techniques will help the situation when perfected. More modeling and computer-aided design will be necessary as complexity grows. It is becoming harder and harder to live without CAD. The major trend in microcomputers will be to move from relatively simple hardware-driven architectures to complex software-oriented structures. We will see full mainframe performance in a microcomputer. Now, however, the software bottleneck restricts how fast electronics gets deployed.

The final observations are based on comments made by John Young. Experimental chips now hold up to half a million components. By the end of the 1980's there will be several million. One of the most exciting computer related areas made practical by VLSI will be artificial intelligence. We are working on removing architectural limitations by building higher-level

functions into hardware and firmware and creating a functional, symbolic interface instead of dealing with details of the hardware structure or with traditional language and operating system interfaces. We must share software more fully so that each user or supplier doesn't waste resources rediscovering proven solutions. The price per megabyte of storage in disks and tapes has been decreasing about 25% a year. In a few more years, we expect processor down time to be measured in thousandths of a percent.

1.1.2 ROTATING MAGNETIC DISKS

Third generation computer systems were highly influenced by the availability of direct access storage devices DASDs,(to use IBM terminology) such as removable media disks and non-removable drums. A removable disk pack would typically have a dozen or more rotating platters with magnetic film on one or both sides. A moving arm actuated by an elecqromagnetic voice coil moves multiple read/write heads to the specific track, of which there are several hundred. The disks rotate at such a speed and the heads move quickly enough to obtain access to any data on the cell (disk pack) within tens of milliseconds. Fixed head disks (non-removable) provide even faster access. As with magnetic tape, advances in storage densities and data transfer rates continue to occur. This subsection reports on the current status or rotating magnetic disk technology and suggests a few improvements yet to come. The price per megabyte for magnetic disks has been decreasing at a rate of 25% per year. Part of this comes from increasing storage capacity without increasing fixed electromechanical costs. Current state of the art in disk storage technology is typified by a recent product announcement by Storage Technology Corporation of their model 8380. This large capacity DASD has two spindles, each spindle with two independent activators. The packing density is about 630 Mbytes per actuator providing a 2.5 Gbyte capacity per unit. It is fully compatible with IBM's 3880. Data transfer rate is 3 Mbyte/second, and average seek time is 16ms. Optional dual port and fixed heads are available. These factors are incorporated into Table 1-8.

Photolithographically produced magnetic recording heads offer a greater linear recording density than do conventional ferrite heads, yet require no alteration in the physical processes of disk recording. The thick film heads consist of layers of materials deposited and shaped by techniques similar to those used in semiconductor technology. The advantages of such a design are:

1. The current required for writing is reduced. Integrated circuits can be used to supply the write current.

2. Manufacturing is simplified. The precision of film photolithography creates well-defined pole-tip geometrics.

3. The head gap is formed independently by a simple photo-resist method.

Table 1-8.

Rotating Magnetic Disk Characteristics

CHARACTERISTICS	VALUE	COMMENT
Storage Capacity	Up to 2.5x 10^9 Bytes per unit	
Access And Transfer Rate	Access Time = 16 milliseconds Transfer Rate = 3 Mbyte per second	
Cost	Medium	Less than semiconductor, greater than tape.
Technological Risk	Low	Rotating disk technology has been commercially available for some time.
Environmental Requirements	Medium	Electromechanical nature of device requires more space and a more controlled environment than other storage media.
Integrity And Robustness	High	Error detection and correction schemes have been developed over the years which enhances system integrity and robustness.

4. The coil can be made with relatively thick, narrow lines to give a low impedance multiturn coil with a good field pattern and edge definition, allowing higher recording densities.

5. The voice-coil activators will reach a technological plateau at about 1000 tracks per inch. Beyond this thin film heads will take over.

6. Greater linear track density and closer tracks give a combined effect of much greater capacity.

1.1.3 OPTICAL ROTATING MEMORIES

Just as magnetic tape has its optical analog, so does magnetic disk. In both situations the nature of the optical recording requires that it be written on only once. Erasable and rewritable materials, such as photochromics tend to have environmental limitations. With the very high recording densities and low cost achievable with current and expected optical recording methods, the write once limitation is not as critical as with lower density storage media.

The May/June 1981 issue of Optical Engineering, guest edited by A. A. Jamberdino of RADC, provides several articles which describe the past and current developments in optical laser recording. Articles by R. F. Kenville, J. W. Watlins et al, and G. J. Ammon provide much of the following material for this subsection. Reports available to U.S. Government agencies can provide the reader with additional information such as RADC-TR-80-46. Less available are surveys by professional technical assessment companies, such as the May 1980 report titled "Impact of Optical Memories (Video Disks) On The Computer And Image Processing Industries," published by Strategic Business Services, Inc.

Kenville, in (Kenville, 81) quotes L. Beiser as stating at the First Annual Wideband Analog Recording Symposium (1967) that laser work had been funded because certain recording requirements could not be met by any other technology. Each has its inherent limitations, although steady progress has been made in approaching these limits. Bandwidths in excess of 100 Mbps and resolutions greater than 10,000 per scan in a practical equipment configuration are theoretically possible according to Kenville. Also reported at the 1967 RADC conference was the concept of ablative optical disk recording. In the early 1970's several companies reported work in laser hole burning of 1 micrometer pits on thin laser sensitive coatings. A disk format was demonstrated for storing and playback of commercial TV frames in 1976.

The optical disk digital recorder/player consists of the rotating disk coated with several layers of laser sensitive and other materials, the laser and its optics and modulator, the tracking and automatic focusing mechanisms, the servo controlled motor, assembly and various electronic control, signal amplifications and conditioning circuits, and the digital interface logic.

Kenville describes the recording and playback operation of a typical optical disk recorder/player as follows:

> "Recording is accomplished by exposing the disk to the modulated radiation from the laser source. This radiation is passed through the modulator and optics and directed towards the disk to form a very small series of holes (determined by modulator inputs) lined up to form a circular track.
>
> During playback, the laser power is reduced by changing its operating mode, and the illumination level to the disk is held constant by inhibiting the modulator. Light reflected from the recorded disk returns back through the focus lens, track mirror, and optics to the data detector. This detector reads out the information previously recorded on the track.
>
> The very short depth of focus of the optical system requires constant repositioning of the focusing lens to keep the laser's spot focused onto the microscopically uneven disk surface. The focus servo accomplished this task by sensing the position for best focus on the disk and driving the focus lens and transducer to maintain this position.
>
> The position of the laser's spot on the disk is determined by the position of the track mirror. This position during record is determined solely by the motor-driven translation stage. The control logic can move this translation stage (on the basis of the input address and mode) so that spiral or circular tracks can be traced. This provides for recording both continuous and single track (one track per disk revolution) information.
>
> During playback, the translation stage is used to determine the approximate portion of the disk that is played back. The translation stage provides control of the laser's spot position on the disk to the nominal track position. As tracks cannot be recorded perfectly circular and some slight disk runout occurs because of

removal and imperfect replacement of the disk, a means of actively locating and following a track must be provided.

Fine tracking control is obtained by a dither track servo that wobbles the track mirror to modulate the position of the readout laser beam on the data detector. The wobbling introduces modulation into the detected return signal, which is used to close the tracking loop and keep the laser beam aligned onto the center of the recorded track. Verifying that the proper track is being followed is accomplished by reading the unique track address recorded with the header information for each track and comparing this to the desired address input to the system." (Kenville, 81).

Watkins, et al, describes the design of an archival mass memory system with a capacity of 10^{13} bits based on proven technology. In addition it is capable of high speed input at a continuous rate up to 50-Mbps. With bit-by-bit data verification one burst is limited to $6x10^{7}$ bits. Multiples of 50-Mbps input can be obtained by incorporating parallel inputs to separate storage units. Each storage unit consists of two sub-units each containing 10^{12} bits of on-line user data and multiple off-line disk packs. Each disk pack contains 1024 optical disks in two levels of 512 each. Disk packs are replaced manually as for a magnetic disk pack. On-line access to any one disk takes less than 15 seconds. Serial equivalent average data transfer rate is 50-Mbps (Watkins, Boudreaux, and Otten, 81).

Ammon describes several variations of archival storage using optical disks. First a high data rate digital recorder/reproducer is being developed. This engineering model is capable of a 320-Mbps rate with a single disk capacity of 10^{11} bits, a maximum access time of 0.5 sec., a bit error rate of less than one in 10^{8} and disk lifetime of ten years. The optical configuration for this using current technology uses a one watt argon laser split into eight individually modulated beams. All beams use a common optical system and form closely spaced tracks on the disk. Ammon suggests that the ultimate solution is believed to be a laser diode array. More development will be necessary to obtain the required power level for recording (Ammon, 81).

Another example of an archival mass memory system, cited by Ammon, requires storage capacities from 10^{14} to 10^{15} bits. This system can have an access time from three to fifteen seconds at a data rate of 50 Mbps and bit error rates of one error per 10^{9} to 10^{10} bits. To achieve this capacity a multiple "juke box" configuration is postulated. Each juke box will contain 100 disks of 10^{11} bits each. Ten juke boxes will be required to obtain 10^{14} bits. An off-line storage facility for 10,000 disks provides the 10^{15} bit total capacity. The disks are stored in cartridges for manually loading into each juke box (Ammon, 81).

Archival capabilities not only require high density and low cost media, but also long life, (e.g., 10 to 15 years) without significant degradation. Tellurium tri-layer disks have been stored in a laboratory environment for two or three years with no noticeable change in signal-to-noise ratio, sensitivity or contrast as of the time of Ammon's article. However, oxidation which will significantly degrade both tellurium and titanium occurs in high temperature, high humidity environments, particularly if not encapsulated in a transparent overcoat.

Ammon compares potential optical disk (OD) characteristics with those of other storage media, such as magnetic disk (MD), computer compatible tape (CCT), and high density magnetic tape (HDMT). First on a projected cost basis, a $10 OD platter capable of storing 10^{11} bits would compare to $100 for HDMT, $1350 for CCT, and $40,000 for MD. The number of storage media units would be, on twelve inch OD, one 2400 foot roll of 2 inch HDMT, 90 2400 foot CCTs, 6250 bpi, and 80 200-Mbyte disk packs, respectively. Furthermore, the high bit packing density makes it easier for optical disks to obtain high data transfer rates, and the disk format provides relatively rapid access to randomly addressed data block (Ammon, 81).

Kenville describes optical disk component technology developments which may contribute to achieving the above design goals and which will provide even greater performance. These include the use of solid state injection lasers, multichannel light modulators, objective lenses with integrated focus/tracking activators, photo detector arrays, and precision turntables. Furthermore, improvements to the storage media will provide longer archival life and eventually reversibility (Kenville, 81).

Optical disk characteristics are summarized in Table 1-9. It appears that the above storage and retrieval technology is progressing dramatically. It may someday outpace our ability to store and process data and to comprehend the information represented therein.

1.1.4 MAGNETIC TAPE STORAGE TECHNIQUES

In this subsection we discuss a storage medium which predates the advent of digital computers -- magnetic tape. As various computer generations have come and gone, the application of magnetic tape to the storage and retrieval of digital information has steadily progressed.

Before direct access storage devices such as drums or disks were available, digital magnetic tape operation became a major design consideration in most batch oriented computer systems. Today, magnetic tape technology is considerably more advanced and appears to be competitive for many types of applications, e.g., serial files, archival storage, high bandwidth recording, etc.

There are basically three groups of magnetic tape devices:

Table 1-9.

Optical Rotating Memory Characteristics

CHARACTERISTICS	VALUE	COMMENT
Storage Capacity	10^{11} to 10^{13} Bits per disk	
Access Time And Transfer Rate	Access Rate = up to 0.5 seconds Transfer Rate = 50 to 320 Mbits per seocnd	
Cost	Medium	Cost per bit is low because of the large capacity of the units; some development is required.
Technological Risk	Medium	Systems are now becoming commercially available but the technology is still somewhat new.
Environmental Requirements	Medium	Reliance on electromechanical devices for manipulating beams is subject to similar constaints as magnetic disks.
Integrity And Robustness	High	Error rates comparable to magnetic tape.

1. Mini/micro computer cassettes and cartridges.

2. Mainframe computer compatible tapes (CCTs).

3. Wideband tapes (WBTs).

In an article appearing in Electronic Design (Yencharis, 81), editor Len Yencharis discusses some trends which he predicts will occur in this decade. Based on estimates made by Control Data Corporation, he states that future recording densities will be greater than 30 Mbits/in^2, as opposed to the current 10 Mbits/in^2. This is close to the theoretical maximum of 1800 tracks per inch and 20,000 bits per inch per track. Another example is the recent announcement by Nortronics of an 18 channel thin film head which they believe will allow 40,000 flux reversals per inch. Yencharis states that further improvements will be made in transfer rates and average access time as well as capacity.

The sheer cost of conversion of existing tape libraries is a major factor in deciding not to go to another mass storage media (Shoor, 80). The magnetic recording industry is continuing to make substantial improvements in cost and bit density and thus presents a moving target to competing technology. Hoagland also contends that the hierarchy-of-memories approach to implement the storage function is here to stay and that magnetic recording will continue to make major contributions through the rest of the century (Hoagland, 79).

To explore where much of the action is we will focus our discussion on computer compatible tapes (CCTs), and on wide-band, high capacity magnetic tape recording (WBTs). A. A. Jamberdino of RADC has recently published an article on high-capacity high-speed recording (Jamberdino, 81). While this article focuses on laser optical techniques, he includes a short review of magnetic recording in his technology assessment for comparison. Jamberdino states that conventional longitudinal storage approaches using IBM compatible formats presently allow total bit capacities of from 2×10^7 to 4×10^7 bits with linear packing densities of 800 and 1600 bits per inch. Recent data acquisition applications have required the development of high density pulse code modulation schemes in which both longitudinal and rotary head recorders are now available with high recording and playback bandwidths. Input/output capabilities of from 80-Mbits per second (Mbps) to 125-Mbps are commercially available. Current packing densities are 30 kilobits per inch on 28-track one-inch tape and storage capacity is 9×10^{10} bits on a single 14 inch reel. Typical bit error rate is 1 error in 10^6 bits without error detection and correction (EDAC) and 1 error in 10^8 to 10^{10} bits with EDAC.

Jamberdino mentions RADC's role in developing high bandwidth, high capacity magnetic tape technology, as well as other high performance storage systems, by citing a number of examples. In 1976, a system was demonstrated which exhibited a 240Mbps bandwidth using 2 inch magnetic tape.

Also proven was the feasibility to provide 30-Kbit/inch/track density with a bit error rate less than one in 10^6, and to provide 64 tracks using a longitudinal format producing a total capacity of $2x10^{11}$ bits per reel. A subsequent effort demonstrated the feasibility of 320-Mbps and recently, Bell and Howell demonstrated 450-Mbps and the potential for 600-Mbps using more tracks at 600 Kbit/inch/track density. This would provide a single reel storage capacity of $7.5x10^{11}$ bits. RCA and Ampex are currently working on helical scan versions capable of 250, 400, and 600-Mbps. This represents a substantial difference from what is commercially available. "Off-the-shelf" magnetic tape "mass storage systems" have data transfer rates about 6 Mbps and storage capacities of 10^{12} bits are achieved by using multiple transports. These are available form IBM, CDC, Ampex, and possibly other firms.

For comparison purposes, we include the capabilities of a write-once laser holographic film approach developed for RADC by Harris Corporation. This technique will accommodate 900-Mbps at a packing density of 1.5 Mbits/cm^2, a bit error rate better than one in 10^6, and a total recording capacity of $1.8x10^{12}$ bits per 10,000 foot reel.

A summary of representative values and ranges for magnetic tape characteristics is contained in Table 1-10.

1.2 Data Base Machines

Advances in hardware such as those described in the previous section, along with the pressing need for more advanced ways of handling large amounts of data, has given rise to a new breed of machines called data base machines. We choose to define data base machines in this context as a hardware/software system which is dedicated to data management functions. This somewhat broad view of data base machines includes several different classes, each of which is discussed in a subsection below:

1. Backend processors
2. Associative memories
3. Cellular logic devices
4. Special purpose function architectures
5. Information storage and retrieval
6. Multiprocessor systems

This taxonomy is based on (Berra, 80).

Table 1-10.

Magnetic Tape Storage Characteristics

CHARACTERISTICS	VALUE	COMMENT
Storage Capacity	14 inch reel = $9x10^{10}$ Bits with multiple transports, up to 10^{12} bits	
Access Time And Transfer Rates	Transfer Rates = 6 to 450 MBits per second	
Cost	Low	Lowest cost storage medium.
Technological Risk	Low	Technology has existed for a long time.
Environmental Requirements	Medium	Tape drive mechanism requires space; equipment is more sensitive than solid state memory.
Integrity And Robustness	High	Numerous error detection schemes have been developed for tape; error rates can be limited to less than one in 10^8.

Data base machines are given the most emphasis in this report because they represent the greatest potential for developing sophisticated data base management and data management capabilities. This potential is based on use of parallelism and advanced, specialized hardware to perform functions rapidly.

1.2.1 BACKEND DATA BASE MACHINES

The purpose of backend data base machines is to off-load data base management functions from an existing mainframe to a directly attached minicomputer. The application programs are executed by the host computer while the backend machine controls access to the data base.

A special communication system provides facilities for the transmission of commands, data, and status information between the host and backend machines. It is responsible for the control of the physical links, proper utilization of line protocols, transmission error and correction processor and task synchronization, and the management of message buffers. Interface routines are established to permit the exchange of information between the host and the backend machines via the communication system.

The host interface, HINT, is the process that is called by the application program when access to the data base is desired. HINT formats the data base request and transmits it as a message to the backend interface through the communication system. HINT later unpacks the message returned from the backend interface and sends it to the application program in the expected format.

The backend, BINT, behaves in a manner analogous to that of the host interface in that it serves as the vehicle for the interchange of information between the communication system and the data base tasks. It unpacks and packs messages that are sent to and from the host interface through the communication system.

There are currently two commercial backend data base machines based on this design. The remainder of this section contains a detailed description of these machines and notes some of the other research efforts in this area.

1.2.1.1 Software AG's Data Base Machine

The first commercial data base machine was developed by Software AG and was called the Data Base Machine (DBM). The DBM combines the firm's Adabas DBMS with an IBM plug-compatible processor from Cambex Corporation. Functioning as a backend processor, the DBM allows users to off-load data base management functions from their mainframes.

The software architecture of the DBM is the well-known Adabas data base management system. The inverted list, multithread system facilitates relational, hierarchical, and network views of the data. Other features available include on-line inquiry, an interactive data dictionary, a programming language interface, a report writer, and restart/recovery utilities.

The "machine" is a dedicated minicomputer called the External Support Processor, ESP. The ESP supports 1M to 8M bytes of main memory and connects to the host through a Channel-to-Channel Communications System (CTCS). It is fully dedicated to data base functions. Most of the Adabas nucleus is off-loaded to the ESP where updates, deletions, additions, and disk management functions are executed independent of the host.

Software AG claims that with the DBM, 60% to 70% of the Adabas CPU time can be off-loaded from the host with an average increase in throughput of 25%. The system sells for around $400,000.

There has been some discussion in the industry as to whether or not the DBM is actually a data base machine. Although DBM now includes an optional query-type nonprocedural language called NATURAL, most programming interaction with DBM occurs in its three lower-level standard procedure languages. The heavily indexed logical structure of Adabas requires an inordinate amount of attention from the host operating system; consequently, some tailoring of the system to the host operating system is necessary to optimize the operating system interface. One view of the system is that part of the existing Adabas software has been transferred from the host to a general purpose minicomputer that can execute Adabas programs concurrently with the host rather than serially. In effect, the DBM is treated as a semi-peripheral connected to a high-speed DMA port with part of the IOCS resident in the peripheral. A table of characteristics describing the DBM is included as Table 1-11.

1.2.1.2 IDM 500

Another example of a backend data base machine is the Intelligent Data Base Machine (IDM 500) manufactured by Britton-Lee. It is a relational data base system with specially designed hardware which has been optimized for the power and flexibility of its relational data base software. The IDM 300 can operate either as a stand-alone data base management system with intelligent terminals or as a backend processor to a host. When operating as a backend processor it significantly reduces the high CPU overhead and memory requirements resulting from the implementation of large software data base management packages.

Data base commands are preprocessed in the front-end system and sent to IDM. IDM buffers the results until the front-end can receive them. The relational data base management system within the IDM can

Table 1-11.

Software AG's Data Base Machine Characteristics

CHARACTERISTICS	VALUE	COMMENT
Storage Capacity	Unknown (Not explicitly mentioned)	Probably can support several large disks.
Addressing Mode	Random Acess	Inverted list, multithread data organization.
Parallelism	Overlapping I/O	
Access Time And Transfer Rate	Medium	No specific rates are given; the basic system does not provide significant acceleration of data access.
Cost	$400,000	Does not include cost of secondary storage.
Technological Risk	Low	The "machine" is a minicomputer and the software is an established DBMS.
Environmental Demands	Medium	Requires IBM plug-compatible minicomputer for front-end, and rotating disks for secondary storage.
Flexibility	High	Allows rational, network, and hierarchical views of the data and provides many system utilities.
Security	Low	Security was not a major design consideration, any security procedures would originate in the host computer.
Integrity And Robustness	Medium	Tailoring of the system to the host operating system could be dangerous.

manage up to fifty data bases, each of which may contain up to 32,000 relations. Each relation may contain up to 2 billion tuples.

Main memory is used both to store the user command and as private workspace for processing. Commands which have been previously defined by the user may be stored in partially processed form and then referenced by a short name or number. This feature helps minimize the transmission time from the front-end to the IDM, and also reduce execution time.

Other features of the system software include the logical data organization which is supported by B-trees and an integrated dictionary. The physical data organization utilizes disk drives and a random access file system. The user of the IDM may control how much space a data base or relation occupies on a disk.

The IDM also has the ability to support a large number of users simultaneously accessing the same or different relations. Its logic controls the processing order of different requests from interfering with one another. One user can have many commands running in parallel on the IDM. User commands are grouped into "transactions". The IDM's crash recovery system guarantees that either the entire transaction will be performed or none of it will be performed. Data base consistency is maintained during power failure, or host processor or IDM failure, by keeping redundant information at certain critical points in the processing of a transaction. In the case of a head crash, it provides the capability of reloading the data base from a checkpoint or dump.

The machine portion of the IDM 500, as shown in Figure 1-3 is organized around a central, very high speed bus. As stated previously, the front-end system can consist of a network of programmable terminals or host computers. The front-end system then communicates with the IDM through I/O processors. There are two types of I/O processors: serial and parallel. Up to eight asynchronous lines can be connected to each serial I/O processor. The parallel I/O processor is multiplexed and can support several front-end systems. Up to eight I/O processors (four serial and four parallel) can be configured in the IDM.

The Data Base Accelerator is an option to the IDM configuration. It is a 10 million instruction per second (MIPS) Schottky TTL pipelined processor with an instruction set specifically designed to support a relational data base. It performs specific functions as directed by the Data Base Processor. The number of transactions/second increases from a rate of two to five transactions/second to a rate of 20 to 25 transactions/second when the Data Base Accelerator is in place. The IDM controller operates with the Data Base Accelerator to process the data at the speed that it is read from the disk. Up to four controller boards may be on the IDM, each of which will support up to four SMD disks. When the Data Base Accelerator is not included in the IDM, the Data Base Processor performs its functions.

The Britton-Lee Intelligent Database Machine

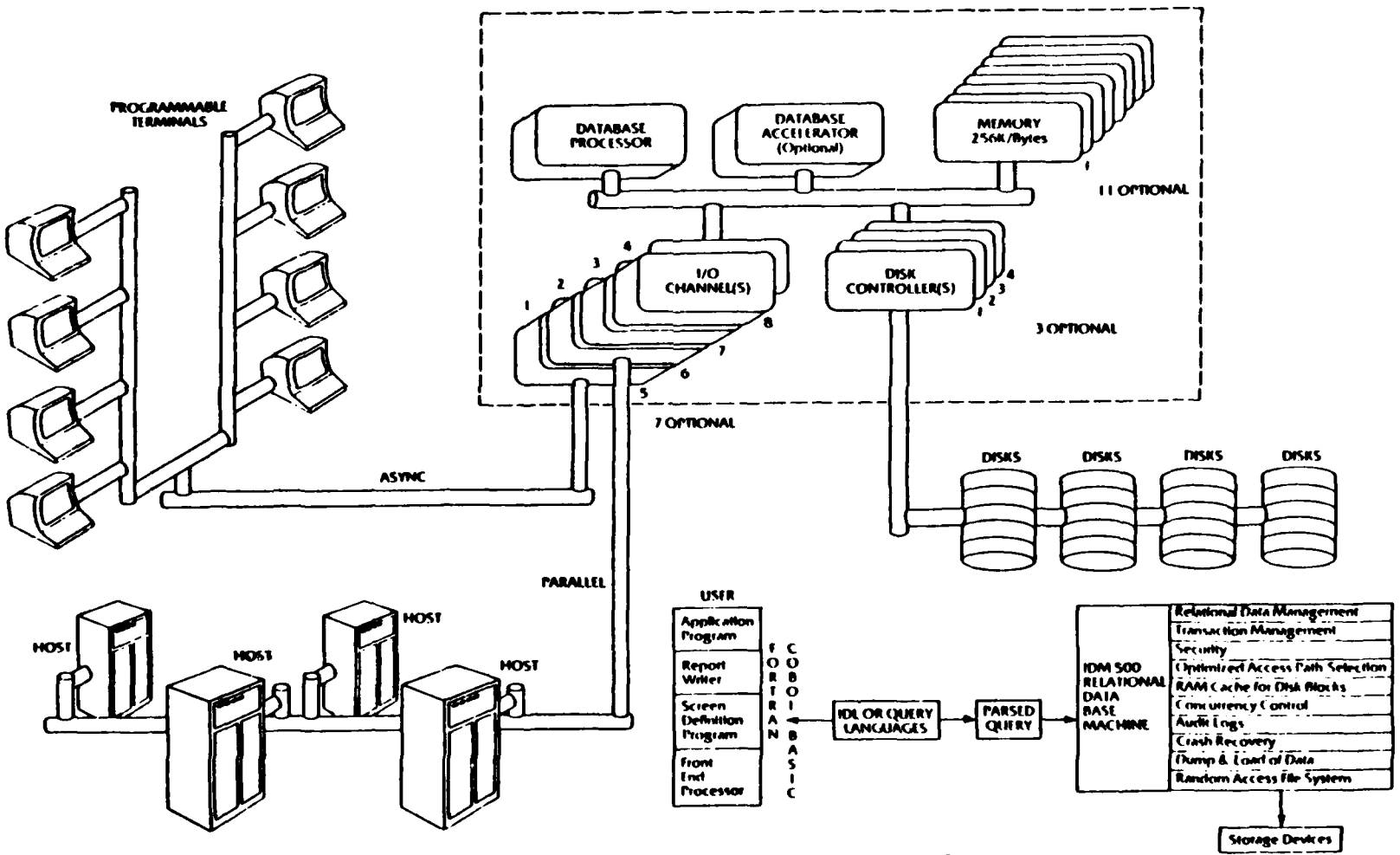


Figure 1-3. Machine Portion of the IDM 500

A RAM Cache is provided in units of 256 Kbytes each; up to twelve of these units can be connected to one system. This cache is used by the IDM to cache pointers, store frequently used disk pages, and to store data associated with user commands. Disk storage is supplied by the user. The maximum storage capacity of all disks is limited to 32 gigabytes.

The IDM accepts commands that have been preprocessed by the front-end system. A user must, therefore, write programs to translate user requests into a form that is understood by the IDM. This translation consists of parsing the user query. A general-purpose query language, IDL (Intelligent Data Base Language) has been developed by Britton-Lee for the purpose of performing these translations into the IDM internal form. It is not necessary to use IDL; it is merely available as a tool and an example for the user.

To round out the general picture of the IDM 500, it supports one clustering index per relation, up to 255 nonclustering indexes per relation, and up to 15 domains per index. The minimum configuration sells for $50,000 for one or $27,00 each for 128 or more. A fully configured IDM 500 can cost over $150,000 with a variable option-dependent discount schedule for quantity purchases. Table 1-12 evaluates the IDM 500 to indicate editing.

1.2.1.3 Research Efforts in Backend Data Base Machine

There have been a number of other research projects which have examined the feasibility of backend processors. The projects described below are surveyed in (Maryanski, 80).

One of the early efforts whose success spawned an enormous amount of follow-on research, was the Bell Laboratories prototype XDMS. A CODASYL-based data base management system, the DMS-1100 was running on the UNIVAC 1108. A tailored CODASYL DBMS, operating system, and communications package were written for the META-4 machine, and a personnel application system was moved to it from the UNIVAC 1108, under DMS-1100, for testing purposes. The XDMS, as the prototype was known, performed successfully on the personnel data base for months. The primary purpose behind the implementation of XDMS was to demonstrate the feasibility of the backend concept. From this viewpoint, SDMS was a complete success.

Inspired by the success of XDMS, the U.S. Army Computer Systems Command sponsored research efforts by the Cullinane Corporation and Kansas State University. The Cullinane effort concentrated on extending IDMS, its IBM 360/370 data base package, to operate in a backend environment.

The IDMS prototype utilized a PDP-11/70 to perform the data base functions for an IBM 370/158. Instead of writing a new operating system, as was done at Bell Laboratories, the vendor-supplied operating system was retained for the IDMS backend. A new version of IDMS, called IDMS-11, was developed for the PDP-11 part of the project.

Table 1-12.

Britton-Lee's IDM 500 Characteristics

CHARACTERISTICS	VALUE	COMMENT
Storage Capacity	Up to 32 gigabytes	
Addressing Mode	Random Access	
Parallelism	Pipelining	10 MIPS Shottky TTL Pipelines processor.
Access Time and Transfer Rate	2 to 5 trans-actions/ second	Increased to 20 to 25 transactions/ second with Data base Accelerator.
Cost	$50,000 Min. $150,000 Max.	Discounts available for quantity purchases. Price does not include secondary storage.
Technological Risk	Medium	Systems have been delivered and are operational but no Data Base Accelerator has yet been delivered.
Environmental Demands	Medium	Requires a network of intelligent terminals or computers as host; relies on rotating disk storage.
Flexibility	High	Flexible relational data base software handles up to 32,000 relations.
Security	Low	Not a major design consideration; relies on host computer.
Integrity and Robustness	Medium	Typical error checking facilities.

The IDMS backend prototype was significant in that it marked the first entrance by a data base vendor into the area of backend systems. Although the first system was developed only to the prototype level, future plans at Cullinane call for the development of a highly secure backend DBMS.

The other backend data base research project supported by the Computer Systems Command was conducted by Kansas State University. The KSU prototypes were developed purely for research purposes. Neither the data base packages, nor the operating systems were altered during the development of the prototypes.

A key difference in this prototype backend DBMS was the use of a large mainframe as a backend to a minicomputer host. The Interdata 8/32 was used as the host with either the ITEL AS5 or the Interdata 7/32 as the backend. Not surprisingly, there was a marked performance difference between the system with the mainframe backend when compared to the system with the minicomputer backend. The ITEL mainframe backend system showed a 41 percent improvement in terms of response time over the Interdata 7/32 system.

A second prototype using only the Interdata machines, but emphasizing modularity and portability over cost and performance, responded from 1.5 to 3 times faster than the first prototype systems. Further research is continuing in the concepts of multiple-processor configurations, the development of high-speed communication links, and the determination of the best languages in which to write interface modules.

The MADMAN system developed by General Electric for execution on PDP-11 hardware differs from the other prototype backend systems in several respects. The most important distinction is the direct connection of the host and backend via shared memory and bus linkages. This architecture provides faster communication links than other backend systems, and simplifies the software required to transmit information between machines.

Another major difference between the MADMAN architecture and that of other prototypes is that only the CPU portion of the data base activity is off-loaded to the backend. Disk operations are performed by the host. This was implemented because primary memory and CPU cycles were deemed to be the critical resources in the MADMAN environment.

MADMAN is the only system with a microprocessor backend. It was designed to be less costly than the other prototypes, portable among several existing PDP-11 minicomputers, and have increased performance over the other systems. Although it appears that the first two goals have been accomplished, more testing is required to determine if the MADMAN design is suitable for the factory control environment.

The basic concept behind associative memories is to access data based on its content rather than its location, and examine the data in parallel. To understand this concept, it is useful to first consider how data is addressed and examined in a conventional sequential computer. One data item at a time is retrieved from memory, based on knowledge of its location in memory. The memory uses an address encoded on the lines of a bus to activate a particular memory cell. With an associative memory, many data items are examined simultaneously, and the data is retrieved based on its content, (i.e., the memory does not handle any address information).

Although there are several different implementations of associative memories, they have a common structure, shown in Figure 1-4. The comparand register contains the value for which the memory is searching. The mast register allows specification of certain parts of the comparand register to be used in the comparison. The memory itself consists of a set of words to be compared to the comparator. The response store typically contains one bit per word, which is set if the word matches the portion of the comparator register being searched for.

An associative memory operates in the following manner. First all the bits in the response store are set. Then, bit by bit, if the appropriate mask register bit is set, all words whose bits do not match the comparator register have their response bits cleared. Such operation is deemed bit slice. Byte slice associative memories perform the comparisons on a byte by byte basis. When the entire word has been shifted through the comparison logic, the response store indicates those words which match the search criteria. Those words then can be retrieved or updated, depending on the application.

There are several advantages of an associative memory. A primary advantage is rapid search of array resident data based on the ability to search by content and in parallel. Another advantage for implementation of relational data bases is that a direct correspondence can be drawn between the relational model of the data and the physical associative memory, since both can be viewed as two-dimensional arrays. A third advantage is that any attribute can function as an index; this is due to the use of any portion of the data word for comparison purposes. Updating can be performed more quickly by using an associative memory: first, location of the item to be deleted or updated can be performed quickly using the search techniques described above; second, since location is not important, additional items can simply be inserted at the end of the list. No pointers need to be updated and data does not have to be physically rearranged. In fact, since pointers or indexes are not required, another advantage, namely lower storage requirements, can also be realized with such an architecture (Berra and Oliver, 79).

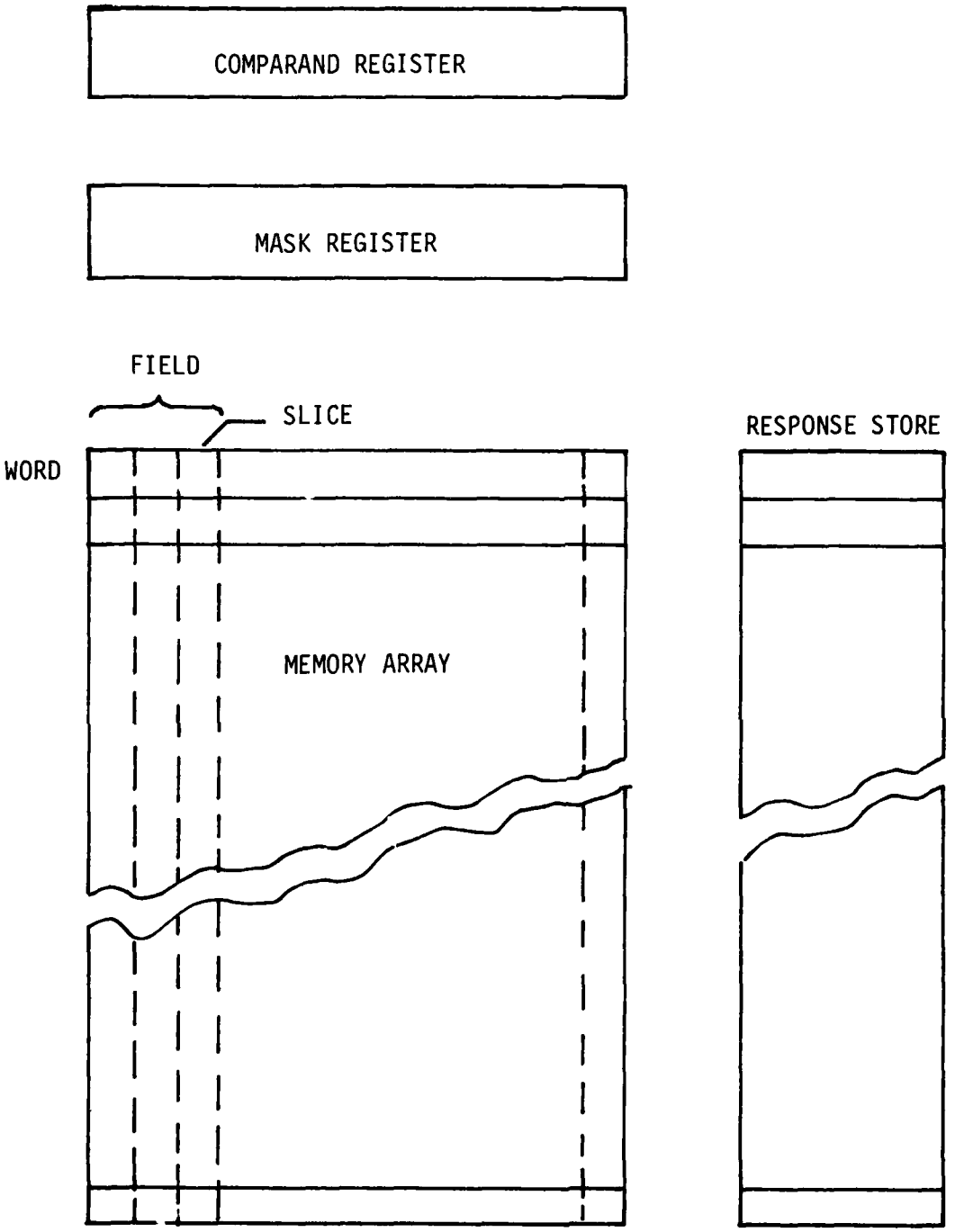


Figure 1-4. Associative Memory Architecture

Despite these advantages, use of associative memories has been largely restricted to research projects. The major problem is that associative memories typically have memory sizes on the order of several megabytes. Actual data bases can easily be several thousand times larger than that. Thus, when searching the data base, the data must be segmented. One segment at a time must be loaded into the associative memory, and the results of each search of a data base segment saved until the entire data base is searched. Unfortunately, the time required for loading and unloading an associative memory can significantly reduce its effectiveness in search operations. Solutions that have been proposed to correct this situation include a high bandwidth interface between the associative memory and a mass storage device containing the data base, or a staging area to bring data in from storage devices. Other less significant limitations are the relatively high price compared to conventional memory hardware, and that each word in the memory must be in a fixed field format. Because of these problems, most of the research involving content addressable memory has been focused on making the associative capability more effective in the context of the overall system.

APPENDIX C

REFERENCES

Ammon, G. J. "Archival Optical Disk Data Storage." Optical Engineering, 1981, 20(3).

Anaker, W. "Computing at 4 Degrees Kelvin." IEEE Spectrum, 1979.

Blum, F. A. "Gigabit Logic Prospects for GADs SLI." Microwave Systems News, 1979.

Burskey, D. "Special Report: Memories Pace Systems Growth." Electronic Design, 1980.

Doty, K. L., Su, U. W., and Greenblatt, J. D. "Magnetic Bubble Memory Architectures for Supporting Associative Searching of Relational Data Bases." IEEE Transactions on Computing, 1980, C-29(1)

Frantz, F. and Norris, D. High Capacity Dynamic Recce Technology. PAR Technology Corporation, RADC-TR-81-369. Griffiss Air Force Base, New York, 1982.

Hnatek, E. R. "Semiconductor Memory Update - Part 2: RAMS." Computer Design, 1980.

Hnatek, E. R. "Semiconductor Memory Update - Part 3: Higher Density Technologies." Computer Design, 1980.

Jamberdino, A. A. Optical Engineering, 1981.

Kadyszewski, R. V. "Trade-off Study of Data Storage Technologies, RCA Final Report NAS5-24170, 1977.

Kenville, R. F. "Fifteen years of Laser Recording - Where We've Been and Where We're Going." Optical Engineering, 1981, 20(3).

Maryanski, F. J. "Backend Data Base Systems." ACM Computing Survey, 1980, 12(1).

Neeno, G. "Serial on Major/Minor Loop? Choose the Bubble Memory to Match The Application." Electronic Design, 1980.

Swanson, P. "Bubble Memories Hold a Lot in Store for UCs." Electronic Design, 1980.

Watkins, J. W., Boudreaux, N. A., and Otten, L. H. "Large Archival Mass Memory System Usting Optical Diskettes." Optical Engineering, 1981, 20(3).

Yencharis, L. Electronic Design, January 8, 1981.

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.